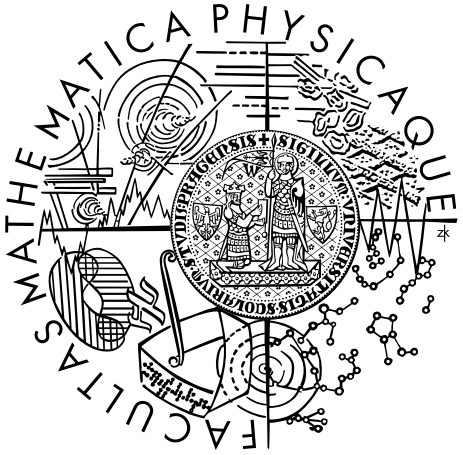
Charles University in Prague
Faculty of Mathematics and Physics

MASTER THESIS



Martin Berger

Guiding a Path Tracer with Local Radiance Estimates

Department of Software and Computer Science Education

Thesis supervisor: Doc. Dr. Alexander Wilkie
Study programme: Software systems
Specialization: Computer graphics

Prague 2011

I would like to thank my supervisor, Doc. Dr. Alexander Wilkie, who assisted me during the creation of the thesis as well as the associated academic papers. His insights and comments proved very useful and contributed greatly to the thesis.

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular the fact that the Charles University in Prague has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 paragraph 1 of the Copyright Act.

In ..... Date ..... Signature

Název práce: Směrování algoritmu sledování světelných cest pomocí lokálních odhadů radiance

Autor: Bc. Martin Berger

Katedra: Kabinet software a výuky informatiky

Vedoucí diplomové práce: Doc. Dr. Alexander Wilkie

Abstrakt: Algoritmus sledování světelných cest (path tracing) je základní, statisticky nestrannou, metodou pro výpočet globálního osvětlení v třírozměrných scénách. Algoritmus je ovšem v praxi příliš pomalý, a proto slouží spíše pro teoretické účely nebo jako základ pokročilejších algoritmů. Tato práce se zabývá určitým vylepšením tohoto algoritmu, kdy při sledování průchodu paprsku scénou algoritmus využívá předpočítaných informací o rozložení světla ve scéně k efektivnějšímu vzorkování možných směrů postupu. Tyto dodatečné informace jsou uloženy v řídké datové struktuře, která je průběžně aktualizována podle potřeby. Algoritmus je implementován v knihovně PBRT.

Klíčová slova: nestranné metody zobrazování, Monte Carlo zobrazování, sférické harmonické funkce, sledování světelných cest, vzorkování podle důležitosti.

Title: Guiding a Path Tracer with Local Radiance Estimates

Author: Bc. Martin Berger

Department: Department of Software and Computer Science Education

Supervisor: Doc. Dr. Alexander Wilkie

Abstract: Path tracing is a basic, statistically unbiased method for calculating the global illumination in 3D scenes. For practical purposes, the algorithm is too slow, so it is used mainly for theoretical purposes or as a base for more advanced algorithms. This thesis explores the possibility of improving this algorithm by augmenting the sampling part, which computes outgoing directions during ray traversal through the scene. This optimization is accomplished by creating a special data structure in a preprocess step, which describes approximate light distribution in the scene and which then aids the sampling process. The presented algorithm is implemented in the PBRT library.

Keywords: unbiased rendering, Monte Carlo rendering, spherical harmonics, path tracing, importance sampling.

Contents

Chapter 1

Introduction

1.1 Rendering

Rendering is undoubtedly a major discipline of computer graphics and starting from the 1970s it is a distinct research area. In a very general sense, it is a means of generating 2D images from 3D geometric scene representations. Typically, we create a mathematical description of the scene - this includes geometric data (shape, position), characteristics of the surfaces of all objects (in essence how the surfaces interact with light), light sources (positions, emission properties) and a sensor (a virtual camera with specified parameters at some location). The renderer then processes all the input data and synthesizes an image of the scene as seen from the sensor.

The first ideas of machine rendering of solids have evolved into a wide variety of renderers, which are used nowadays. They differ greatly in their capabilities and they find their applications in many commercial and academic fields.

1.2 Motivation

A major incentive to rendering research comes from the entertainment industry. Most movies use some kind of visual effects that are artificially created on a computer. They can either be blended with live footage or create completely artificial shots, but in both cases the renderings must be fairly accurate to avoid mismatches between the generated and filmed parts. A global illumination solution is a necessity for these purposes, since otherwise the human sight would immediately detect inconsistencies and the illusion of the effects would be lost.

Not every movie producer seeks to make only photorealistic looking movies, though. Examples are cartoon-like movies or animated movies for children. A different branch of rendering is employed here - non-photorealistic rendering. The renderers do not need to exactly obey the physical laws of light transport here, but they must still keep their output consistent with at least the basic light propagation rules that a human viewer assumes.

Computer games are another example of the application of renderers. Although the ultimate goal of realism is the same as with movie effects, the requirement of interactivity makes the problem much harder. Instead of hours or days, the time available for rendering a single frame is of the order of milliseconds. Even with the power of today's massively parallelized graphical processing units

(GPUs), radical simplifications need to be done to make such rendering possible. It is usually not a big problem, because gameplay is more important than visual realism in the gaming context.

Other important areas of realistic image synthesis are in design and prototyping. A prominent example is architecture, where visualizations of both the interiors and exteriors are usually required long before the actual construction begins. Here, physically accurate results are more important than anywhere else - the visualizations are used to evaluate and compare different designs prior to their actual existence. Every link of the image synthesis chain is equally important here, so not only the actual rendering, but also for example the material properties, light emission characteristics or image post-processing have to be handled carefully.

1.3 Challenges of realistic rendering

An interesting characteristic of each algorithm is the time to render a typical image. It can generally be in the order of milliseconds to days for a single image. Although some rendering algorithms may be hard to classify according to the rendering time criterion, we can divide them in two large groups: real-time and non-real-time. In our work, we focus solely on non-real-time rendering, where we want to get as close to the physical model of real light behavior as possible.

Even if we restrict ourselves to non-real-time renderers, there are numerous differences in the features these renderers provide. In general, each renderer poses a trade-off between realism and rendering speed. In this thesis, we are concerned with renderers capable of maximal realism. They need to encompass all aspects of how light travels through a scene before reaching a sensor. These especially include:

- *Scattering of light at surfaces*: This involves simulating how light interacts with material interfaces. A light ray upon hitting a medium interface can reflect itself, refract into the medium, be absorbed, or can undergo a combination of all three cases. The exact physical processes that determine the amount and direction of light after the interaction can be very complex for real materials. Therefore, the common approach here is to physically and mathematically describe a simplified situation with a flat homogeneous interface and model the more complex surfaces stochastically.

- *Illumination*: The renderer must be able to determine how the light from light sources is distributed in the scene. This leads to the challenging problem of how to compute the contribution of *all* possible light paths of any length between light sources and the receptor. There are several effects known for their difficulty hidden in this general description, for example accurate soft shadows or caustics - places where light concentrates behind highly specular objects.

- *Effects related to the sensor*: When simulating real world sensors, such as human eyes or a digital camera, we can no longer consider parameters like shutter and aperture as infinitesimal quantities. The related effects include

depth of field (blurry appearance of out of focus objects) or motion blur (blurry appearance of moving objects).

- *Participating media*: A common assumption is that the light rays travel through vacuum between their bounces on the scene surfaces. However, this assumption rules out simulation of scenes with fog or thick translucent objects such as wax, a glass of milk and so on. An accurate renderer must account for this by computing how light scatters during transmission through non-clear media.

Some of these effects are shown in figure 1.1. Depending on the requirements, additional effects might be necessary to deal with, for example light polarization, diffraction, fluorescence or dispersion.

The most accurate renderers proceed by completely solving the so-called rendering equation (introduced and discussed in chapter 2), which serves as a mathematical model of physical light transport in a scene. Solving this equation naturally captures most of the real world optical phenomena given above.

Examples of such algorithms are the path tracing algorithm or Metropolis light transport. Both of them achieve photorealistic results, but they often require an enormous amount of time to render a single noise-free image. Therefore, it is important to analyze all possibilities to improve these algorithms - even a minor improvement might result in huge rendering time savings.

Figure 1.1: A photograph of a real world setting showing complex lighting effects which are hard to simulate by traditional rendering algorithms. Note especially the caustics - bright spots of light focused by glass surfaces.

1.4 The goals of the thesis

The main goals of this thesis are:

- Analyze the possibility of augmenting the path tracing algorithm with a separate data structure that contains an approximate distribution of light in the rendered scene. The introduction of this kind of information should enable focusing of the rays to directions with significant light contribution and in effect increase the speed of convergence. The thesis should examine present approaches and suggest a suitable data structure to hold this information along with a robust and fast method of utilizing it.

- Implement the proposed method in the PBRT rendering research framework [PH10]. Compare the performance to other rendering algorithms and describe situations where the new algorithm offers superior performance as well as situations where it does not offer any significant advantages. The new method should be evaluated with respect to performance, ease of implementation and applicable scenarios.

- Discuss the implications of the modifications of the path tracing algorithm on the unbiased nature of the whole rendering scheme.

Chapter 2

Unbiased Monte Carlo rendering

2.1 Overview

The roots of unbiased rendering date back to the work of Kajiya [Kaj86], who was the first to publish the famous rendering equation which describes the complete light transport in a closed scene. Since it is an approximation to the Maxwell's equations that describe electromagnetism, it serves as an important generalization of all rendering algorithms which simulate real world light propagation. Kajiya also summarized all major rendering algorithms which had been published before along with an analysis of how these algorithms approximate the solution to the rendering equation. Of all these algorithms, distribution ray tracing [CPC84] provided the most accurate results, but Kajiya himself recognized the prohibitive inefficiency of the idea behind it (extensive numerical integration that leads to a high fan-out at each recursion level).

2.2 The rendering equation

The original rendering equation from Kajiya's work is parametrized by two arbitrary points in the scene and gives the radiance that flows between these two points. However, it is often useful to rewrite the equation in some other, equivalent way. In our work, we will often use the rendering equation parametrized by scene point and outgoing direction (the equivalence of the parametrizations is shown for example in [DBB06] or in [PH10], pp. 754-757):

$$L(\vec{x},\vec{\omega}) = L_e(\vec{x},\vec{\omega}) + \int_{\Omega_i} f_r(\vec{x},\vec{\omega_i} \to \vec{\omega}) L_i(\vec{x},\vec{\omega_i}) \cos\theta_i d\omega_i, \tag{2.1}$$

where

- $L(\vec{x},\vec{\omega})$ is radiance coming from point $\vec{x}$ in the direction $\vec{\omega}$. Mathematically, directions are represented by infinitesimal solid angles.
- $L_i(\vec{x},\vec{\omega_i})$ is radiance coming to point $\vec{x}$ from the direction $\vec{\omega_i}$.
- $L_e(\vec{x},\vec{\omega})$ is radiance emitted from point $\vec{x}$ in the direction $\vec{\omega}$.
- $f_r(\vec{x},\vec{\omega_i} \to \vec{\omega})$ is the so called bidirectional reflectance distribution function (BRDF). For a given point $\vec{x}$, it determines how much light coming

from direction $\vec{\omega_i}$ is reflected into direction $\vec{\omega}$. This function captures material properties, i.e. how light interacts with a given interface. If this function complies with some reasonable physical constraints (namely Helmholtz reciprocity, positivity and the law of energy conservation, for details see [DBB06], pp. 32-35) and if we fix the incoming direction, it can be viewed as a bivariate probability density function (PDF) for the reflected direction.

- θ_i is the angle between surface normal at $\vec{x}$ and the incoming direction ω_i.
- Ω_i denotes a hemisphere of incoming directions ω_i centered around surface normal at $\vec{x}$.

Another equivalent form, which integrates over the surface area instead of the outgoing direction, is given by:

$$L(\vec{x}, \vec{\omega}) = L_e(\vec{x}, \vec{\omega}) + \int_A f_r(\vec{x}, \vec{\omega_i}(\vec{x}, \vec{x'}) \to \vec{\omega}) L_i(\vec{x}, \vec{\omega_i}(\vec{x}, \vec{x'})) G(\vec{x}, \vec{x'}) dA, \qquad (2.2)$$

where

- $G(\vec{x}, \vec{x'})$ is a geometric term that arises from the change of variables. It also includes a visibility term that evaluates to 0 or 1 depending on whether $\vec{x}$ is directly visible from $\vec{x'}$.

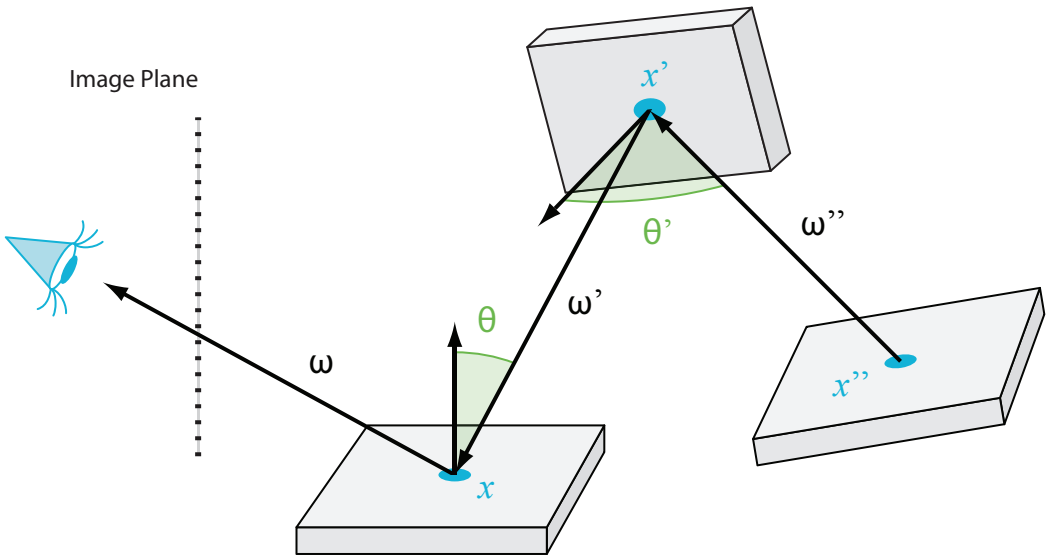


Figure 2.1: An explanation of the terms used in the rendering equation.

This type of equation is known as the Fredholm integral equation of the second kind, since the unknown L appears also on the right hand side of the equation inside the integral. Note that there also exist generalizations of this equation which take into account for example time or wavelength of the light. These parameters are unrelated to the presented ideas, so they are not included for clarity.

2.3 Path tracing

In the same paper, Kajiya also proposed a rather simple, ray-tracing based Monte Carlo algorithm that solves the rendering equation. This algorithm is now known as the path tracing algorithm and computes the radiance estimate $\hat{L}(\vec{x}, \omega)$ from independent random light paths through the scene that end up in the point $\vec{x}$ and in the direction ω. According to the Monte Carlo theory, the estimate is computed as

$$\hat{L}(\vec{x}, \omega) = \frac{1}{N} \sum_{n=1}^{N} f(X_n), \tag{2.3}$$

where X_n are samples from the space of light paths (sampled randomly with uniform distribution) and $f(X_n)$ denotes a recursive evaluation of the right hand side of equation 2.1 up to the depth determined by the length of the path X_n. The light paths are constructed as follows (see also figure 2.2):

1. Set $w = 1$.
2. If the surface at point $\vec{x}$ emits nonzero radiance L_e in the direction ω, add $w * L_e(\vec{x}, \omega)$ to the result.
3. Choose randomly an outgoing direction ω_o and set

   $$w = w f_r(\vec{x}, \omega_o \to \omega) \cos \theta_o.$$

   Trace a ray to find a new intersection point $\vec{x}$, set $\omega = -\omega_o$ and go to Step 2.

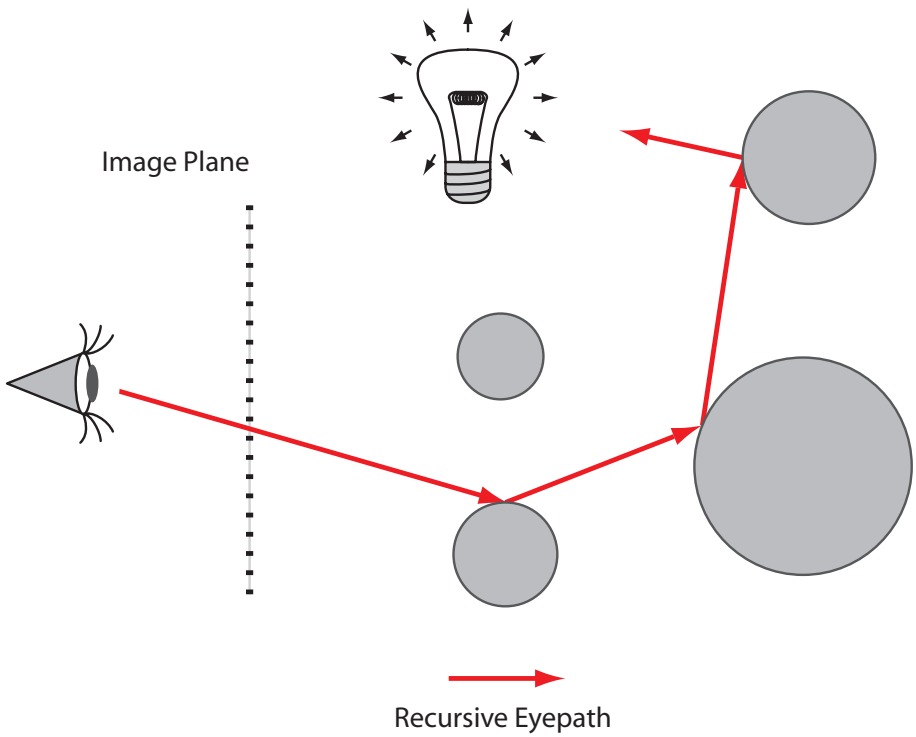


Figure 2.2: A schematic visualization of the path tracing algorithm.

To keep the Monte Carlo estimator 2.3 unbiased, the contributions of the light paths must be correctly weighted by the corresponding probabilities [Vea98], pp. 37-38. These can be derived using Markov Chain theory (see [Kaj86] for details) to be the products of the outgoing ray probabilities at each bounce.

To see that the constructed light paths indeed sample the integrated function, we can express equation 2.2 in a compact manner using integral operator notation:

$$L = L_e + \mathcal{T}L,$$

where

$$(\mathcal{T}L)(\vec{x}, \vec{\omega}) = \int_A f_r(\vec{x}, \vec{\omega_i}(\vec{x}, \vec{x'}) \to \vec{\omega}) L_i(\vec{x'}, \vec{\omega_i}(\vec{x}, \vec{x'})) G(\vec{x}, \vec{x'}) dA(\vec{x'}).$$

A formal solution can be obtained by iteration:

$$L = L_e + \mathcal{T}L = L_e + \mathcal{T}(L_e + \mathcal{T}L) = L_e + \mathcal{T}L_e + \mathcal{T}^2 L = \ldots = \sum_{n=0}^{\infty} \mathcal{T}^n L_e.$$

The resulting sum is called the Neumann series and its convergence is assured by the fact that the amount of radiance decreases at each reflection ([CBNR11]). If we now expand one summand (let us say n-th), we get:

$$\mathcal{T}^n L_e = \int_A \ldots \int_A \quad f_r(\vec{x_0}, \vec{x_1}, \vec{x_2}) \cdots f_r(\vec{x_{n-1}}, \vec{x_n}, \vec{x_{n+1}}) G(\vec{x_0}, \vec{x_1}) \cdots \\ G(\vec{x_{n-1}}, \vec{x_n}) L_e(\vec{x_n}, \omega(\vec{x_n}, \vec{x_{n-1}})) dA(\vec{x_0}) \cdots dA(\vec{x_n}) \tag{2.4}$$

This equation may look intimidating at first glance, but it can be explained intuitively. It tells us that the n-th term of the Neumann series contains the contribution of all possible light paths exactly n bounces long. The bounces occur at the points $\vec{x_1}, \vec{x_2}, \ldots, \vec{x_{n-1}}$. The whole sum then contains all light paths of arbitrary length and this shows the connection to the sampling algorithm given above.

2.4 Terminating criterion

As outlined, this algorithm still needs some stopping criterion to guarantee that it will terminate. A naive approach is to limit the recursion depth by some constant. While it certainly makes the algorithm terminate in finite time, it introduces bias (nonzero expected value of the difference between the computed and real solution) to the computed values. This is because it completely discards a set of light paths that, although probably insignificantly, still should contribute to the final result.

A slightly better approach is to keep track of the product of all BRDF and cosine terms of all bounces along the ray path being generated and terminate the ray if this product drops below some threshold. This technique is more efficient than the previous one, because it does not blindly discard paths of certain length. Imagine a ray that starts with three bounces on mirror surfaces and then hits a diffuse surface. This can be potentially a very important light path,

because the mirror bounces do not attenuate the light at all. With fixed maximal number of bounces, we would miss these types of path, whereas the accumulated multiplication factor method would not. It will still introduce bias to the solution, though, because it systematically discards light paths with small, but nonzero contribution.

A better solution is to use a neat probabilistic trick called the Russian roulette, which can terminate the path at any bounce with some probability P. At each bounce, we draw a sample s from the uniform distribution over $[0, 1]$ and terminate the ray, if the value is below P. Then, we modify the original estimator $f(X_n)$ to account for the samples that were killed:

$$f'(X_n) = \begin{cases} \frac{f(x_n)}{1-P} & \text{if } s > P \\ 0 & \text{otherwise.} \end{cases} \tag{2.5}$$

The estimator remains unbiased, because the expected value does not change:

$$\mathbb{E}f' = (1-P)(\frac{\mathbb{E}f}{1-P}) + P \cdot 0 = \mathbb{E}f. \tag{2.6}$$

The main drawback of this technique is that it increases variance. The severity depends on the probability P, since:

$$\begin{aligned} var(f') &= \mathbb{E}(f'^2) - (\mathbb{E}f')^2 \\ &= (1-P)\mathbb{E}(\frac{f}{1-P})^2 + P \cdot 0^2 - (\mathbb{E}f)^2 \\ &= \frac{\mathbb{E}f^2}{1-P} - (\mathbb{E}f)^2 \\ &= (\frac{1}{1-P} - 1)\mathbb{E}f^2 + var(f). \end{aligned}$$

From this relation we immediately see that any nonzero value of P will increase the variance and that the value should be chosen as small as possible. There is no need to set the value constantly for all possible bounces. Indeed, the most commonly used value for the probability of termination is the surface hemispherical reflectance, or albedo [DBB06], pp.113-114. This follows the physical behavior of light, which is more easily absorbed on dark surfaces.

2.5 Complexity and optimizations

The equation 2.4 also shows the complexity of the problem. To compute only the n-th Neumann series summand, we need to integrate over $2n$-dimensional space. Furthermore, the integrated function is far from trivial and it is discontinuous (the visibility part of the geometric terms for example), so it becomes clear that Monte Carlo integration is the only viable option. It can handle integration of multi-dimensional (even infinite-dimensional) integrals of discontinuous functions and relies only on sampling and integrand evaluation. The rate of convergence is $\mathcal{O}(N^{-1/2})$ regardless of the dimensionality of the integral, which means that

doubling the number of samples reduces the root mean square error (RMSE) σ to $\frac{\sigma}{\sqrt{2}}$ (for a proof see [Vea98], pp. 39-40).

Albeit valid and unbiased, the presented algorithm is horribly inefficient in typical scenarios. Shooting the rays completely at random at every bounce is a huge source of inefficiency, since it disregards all information about material properties and light sources, which are known to the renderer.

One of the easiest optimizations is to split the integrand into direct and indirect lighting parts (next event estimation). While this is mathematically a trivial modification (additivity of integration), it can be really powerful in practice. At each bounce two rays are sent forth (see figure 2.3) - one that collects light emitted directly by the light sources towards the point in question and a second one which collects only indirect lighting (disregards direct light emission at the next intersection). The direct lighting ray traversals are terminated immediately at their first intersections, so it may be beneficial in some applications to send more than one lighting ray to speed up convergence. The indirect lighting ray continues the scene traversal exactly like in the original path tracing algorithm.

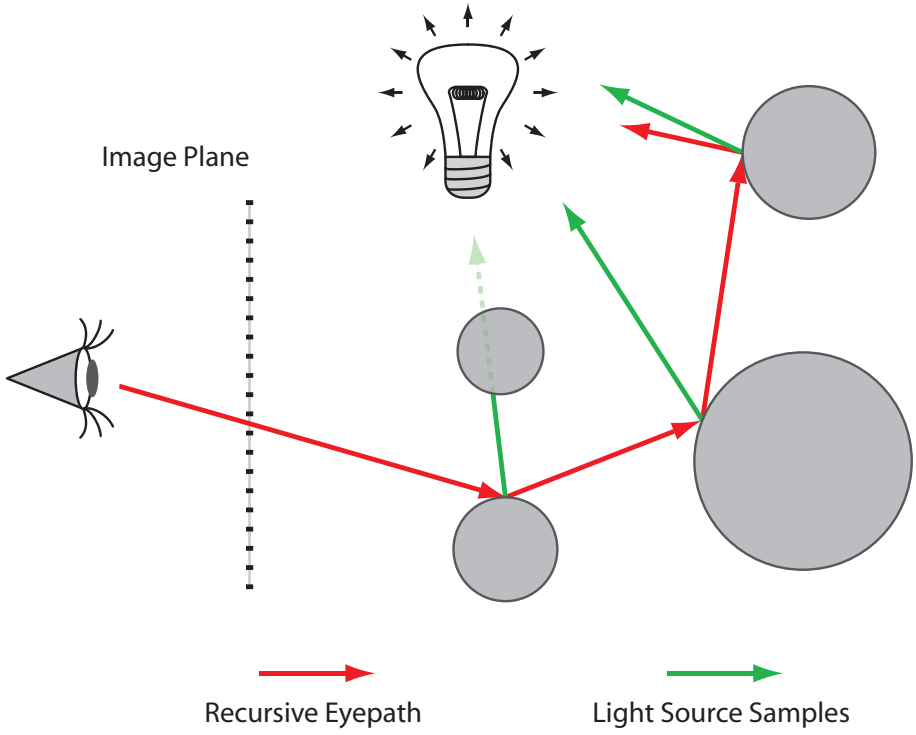


Figure 2.3: A schematic visualization of the path tracing algorithm with explicit light source sampling.

Monte Carlo theory offers further possibilities to reduce variance of the estimator, for example importance sampling, control variates or stratified sampling, to name a few. A comprehensive discussion in the context of computer graphics can be found in [Vea98], pp. 45-71. For this work, importance sampling theory is crucial, so it is useful to write about it in more detail here.

2.6 Importance sampling

The evaluated integral may have some regions which have much higher function values than other regions. These regions are much more important to sample because such samples influence the total estimate greatly. The rendering equation (2.1) itself contains two such examples - BRDF and incoming radiance. Let us look more closely at these two examples. The incoming radiance function, which is parametrized by incoming direction, can contain bright spots such as light sources that contribute significantly to the total irradiance. On the other hand there might be other regions that are completely insignificant. In the case of BRDF, an ideal example is a highly specular material. Only a narrow lobe around the reflected ray contains directions that will contribute to the result. It is clear now, that if we take uniform samples, we are likely to miss or undersample these important regions, so it could be beneficial to draw the samples according to a PDF proportional to the integrated function.

Let us denote the integrated function f. The theory tells us that if we use a non-uniform distribution to obtain the samples, our Monte Carlo estimator changes to:

$$\hat{L}(\vec{x}, \omega) = \frac{1}{N} \sum_{n=1}^{N} \frac{f(X_n)}{p_{X_n}}, \tag{2.7}$$

where p_{X_n} is the probability of taking the sample X_n. Another view on this operation is that we are basically transforming the variable in such a way that the integrated function becomes more uniform as a result. A uniform function can be handled much better by standard uniform Monte Carlo integration.

An optimal choice of the distribution would be $p(X) = \frac{f(X)}{\int_\Omega f(X)}$, which would cause the variance to vanish completely. Unfortunately, this choice poses two serious problems - we would need to know the value of the integral, which is of course the value we want to compute in the first place and we do not have any general method to sample arbitrary functions other than rejection sampling, which could severely affect the performance. Therefore, we have to resort to choosing a distribution that at least approximates the original function and which can be sampled effectively.

In the case of direct lighting rays, two basic strategies exist: light sources sampling and BRDF sampling. Light sources sampling tries to send the rays towards the light sources in hope that they are visible and will contribute to the result. However, it does not take into account the BRDF of the material, which results in poor performance for example if the surface is highly specular. Even if the rays hit a light source, they can be completely downweighted by the BRDF term and thus rendered useless.

On the other hand, BRDF sampling guarantees that rays are sent only in directions which can contribute to the result. But this approach also has its deficiency - it will miss for example small light sources, because it is highly improbable to hit them without explicit sampling.

Looking at these two approaches, it seems natural to try to combine them in some way, because precisely those scenarios that are handled well by one sampling method are problematic for the second one and vice versa. Unfortunately, it is not immediately clear how to take results from both estimators and combine them

properly. This problem was solved by Veach et al. [VG95] who describe a method to sum results of several estimators that remains unbiased. It is called multiple importance sampling (MIS).

As for the indirect lighting ray, only BRDF sampling is commonly used, because the complete incoming light distribution is unknown to the renderer. Note the difference to the previous case of direct lighting, where the knowledge of the light sources properties was sufficient. To get the incoming light distribution in the indirect case, we would need to solve the rendering equation, so the only possibility is to get at least some estimates of the incoming radiances. A first approach to this subject is by Jensen [Jen95] and it is also this thesis, where we propose a rendering algorithm that uses this kind of sampling.

2.7 Other algorithms

The path tracing algorithm, as outlined in previous sections, is an example of gathering solvers, which means it shoots rays from the receptor into the scene towards the lights and accumulates, or gathers, light. This process is in fact contrary to what could be intuitively expected and what would correspond to real world light propagation, where light travels from the light sources towards the receptor, where it is absorbed. As it turns out, there is no mathematical reason to prefer one way or the other and both approaches are equivalent [PM95]. The rendering equation can be accordingly rewritten to create an adjoint equation called the potential equation and an algorithm similar to path tracing, in this case named light tracing, can be used to solve it. It proceeds by shooting light particles from the light sources and tracing them through the scene until they hit the sensor's aperture. Although valid and unbiased, it is typically much less efficient than path tracing, because few light particles ever hit the sensor before they are either killed by Russian roulette or attenuated by too many bounces. This leads to it not being used alone, but rather as a part of other, more sophisticated, algorithms.

Bi-directional path tracing [LW93] (BDPT), a hybrid path-based gathering technique, combines path tracing and light tracing into one unbiased algorithm. It starts by tracing a ray from the sensor and a second one from the light sources. After sufficient number of bounces it connects all samples on these two paths and creates many light-eye channels that contribute to the final result. Although it can handle complex lighting scenarios, it has serious performance issues, since it causes a very high ray-casting overhead to connect all light-path and sensor-path samples. This has led to it not being widely used, since ray-casting operations are amongst the most costly operations in a rendering system.

A highly important, related ray-based technique is Metropolis Light Transport [VG97]. This technique extends bi-directional path tracing, and increases its efficiency by attempting to re-use paths that have proven to be valuable. Once the BDPT finds a light channel that non-trivially contributes to the computed result, a series of probabilistic perturbations is applied to the light path to explore the nearby space, which is supposed to be just as important due to coherency. This technique excels at solving "difficult" lighting scenarios, but introduces a considerable algorithmic overhead over an already complex BDPT engine that is used as the base for the algorithm. Consequently, there can be cases (such as

scenes with straightforward and easy to solve lighting arrangements) where it is actually less efficient than MIS path tracing.

There are also a large number of techniques that stochastically shoot energy from the light sources into the scene, and which then use this information for rendering purposes. Photon Mapping [Jen96] was the first such technique, but the concept has been extended numerous times since then [HOJ08, HJ09]. Common to all techniques that directly use information from Photon Maps, or any other averaging data structure that records energy shot into the scene, is that such techniques cannot be reliably unbiased any more due to the discretization and reconstruction errors introduced through the recording data structure. If an unbiased renderer is desired, Photon Maps and the like can be used – but only to guide the sampling process, and never as a direct source of information. In this thesis, we go in exactly this direction.

Chapter 3

Spherical harmonics and importance sampling

The algorithm presented in the thesis relies on storing its radiance estimates as coefficients in spherical harmonics basis. It is therefore essential to introduce the mathematical background and discuss methods used to importance sample such functions along with an analysis of the associated problems. After the review of the present work, a robust importance sampling approach suitable for unbiased Monte Carlo rendering is derived and presented.

Parts of this chapter are strongly based on [Ber11], where we presented some insights on this topic.

3.1 Overview of spherical harmonics

Spherical harmonics are a set of functions $y_l^m(\theta, \phi)$ which form an orthogonal basis of square-integrable functions defined over the spherical domain. Thus, any such function can be represented as a series of coefficients in this basis.

Depending on the error we can tolerate, a relatively complex function can be stored as just a few floating-point numbers. This is a key advantage over other representations such as sampling and tabulating the values, where a lot more data would need to be stored.

3.2 Mathematical definitions

Real spherical harmonic basis functions are defined by:

$$y_l^m(\theta, \phi) = \begin{cases} K_l^m P_l^{|m|}(\cos\theta) \cos|m|\phi, & \text{for } m \geq 0 \\ K_l^m P_l^{|m|}(\cos\theta) \sin|m|\phi, & \text{for } m < 0, \end{cases} \tag{3.1}$$

where $l \in \mathbb{N}_0$, $m \in \{-l, -l+1, \ldots, l\}$, K_l^m are constants and P_l^m are the associated Legendre polynomials given by:

$$P_l^m(x) = \frac{(-1)^m}{2^l l!} \sqrt{(1-x^2)^m} \frac{d^{l+m}}{dx^{l+m}} (x^2 - 1)^l. \tag{3.2}$$

The normalization constants K_l^m are chosen as:

$$K_l^m = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}. \tag{3.3}$$

so that the set of all functions $y_l^m(\theta, \phi)$ forms an orthonormal basis of the space of square integrable functions defined on the surface of a sphere. Specifically, the orthonormality can be written as:

$$\iint_{4\pi} Y_l^m Y_{l'}^{m'} = \begin{cases} 1 & \text{if } m = m' \text{ and } l = l' \\ 0 & \text{otherwise.} \end{cases} \tag{3.4}$$

For a proof of orthonormality and a more detailed description of spherical harmonics and their properties, see [Slo08] or [Gre03]. Figure 3.1 shows basis functions corresponding to the first three bands ($l = 0, 1, 2$).

For practical purposes, the formal definition 3.2 is very impractical and numerically unstable. The most stable and efficient way to evaluate spherical harmonics is to use recurrence formulas [PTVF92] for the associated Legendre polynomials

$$\begin{aligned} P_0^0(x) &= 1 \\ P_1^0(x) &= x \\ P_m^m(x) &= (-1)^m(2m-1)!!(1-x^2)^{m/2} \\ P_{m+1}^m(x) &= x(2m+1)P_m^m(x) \\ P_l^m(x) &= \frac{x(2l-1)P_{l-1}^m(x) - (l+m-1)P_{l-2}^m(x)}{l-m} \end{aligned}$$

and plug these values into 3.1.

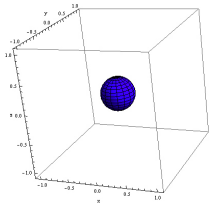
Let us now consider a square integrable function f defined on the spherical domain. Due to orthogonality, the coefficients of this function projected onto the spherical harmonic basis can be obtained from:

$$c_l^m = \iint_{4\pi} f(\theta, \phi) y_l^m(\theta, \phi)\, d\theta\, d\phi \tag{3.5}$$

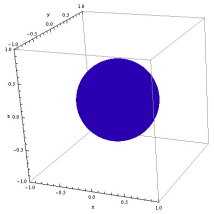
In the case of a band-unlimited function f (for example when f contains a discontinuity), the projection step will yield an infinite number of nonzero coefficients. For practical purposes, we truncate the series by setting $y_l^m = 0$ for $l > N$, where N is a pre-determined maximum band. In effect, we band-limit the function and discard some high frequency content. During reconstruction, we approximate the original function by summing the basis functions weighted by the coefficients c_l^m:

$$f(\theta, \phi) \approx \hat{f}(\theta, \phi) = \sum_{l=0}^{N} \sum_{m=-l}^{l} c_l^m y_l^m(\theta, \phi). \tag{3.6}$$

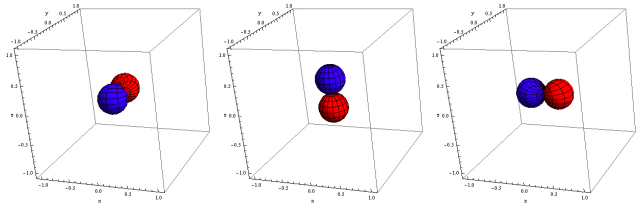
Let us suppose now that we have two functions f and g represented by SH coefficients f_l^m and g_l^m, and we want to obtain coefficients of the function $\alpha f + \beta g$.



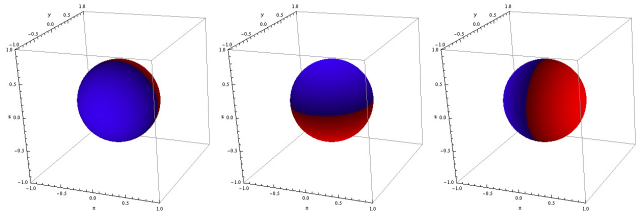
(a) $l = 0, m = 0$.



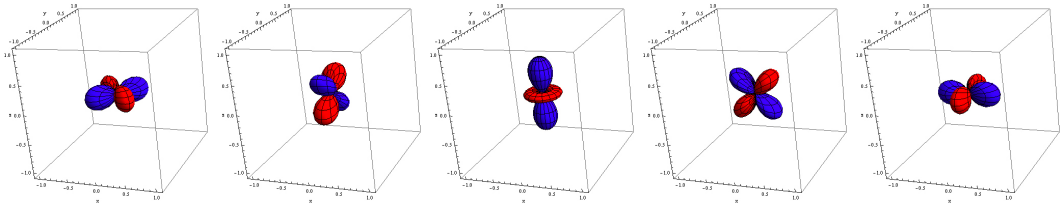
(b) $l = 0, m = 0$, unit sphere distribution.



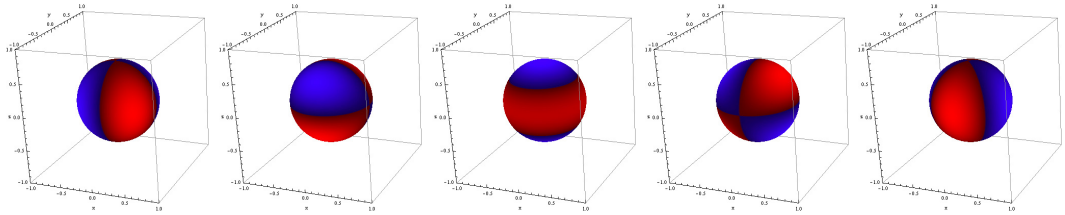
(c) $l = 1, m = -1, 0, 1$.



(d) $l = 1, m = -1, 0, 1$, unit sphere distribution.



(e) $l = 2, m = -2, -1, 0, 1, 2$.



(f) $l = 2, m = -2, -1, 0, 1, 2$, unit sphere distribution.

Figure 3.1: The first three bands of the spherical harmonics basis functions. Two visualizations are used - spherical plots, where the function magnitude is shown as distance from the origin, and distributions on unit spheres. In both cases, blue color corresponds to positive function values and red color to negative values.

Looking at equation 3.5 and considering additivity of integration, we immediately get

$$(\alpha f + \beta g)_l^m = \alpha f_l^m + \beta g_l^m. \tag{3.7}$$

This relation is of great importance for this work, since it shows that the spherical harmonics coefficient vectors can be component-wise interpolated and this operation is equivalent to interpolation between the functions in the spatial domain.

It may be tempting to draw a similar conclusion about the product fg, but obtaining the correct coefficients is more difficult in this case. Again, from 3.5 we get:

$$(fg)_k = \sum_{i,j} (\int y_i(\omega) y_j(\omega) y_k(\omega) \mathrm{d}\omega) f_i g_j, \tag{3.8}$$

where the double indices for each basis function were reindexed as i, j as k for clarity. The quantity inside the parentheses is called a triple product tensor and can be precomputed since it depends only on the SH basis functions. However, for increasing number of bands, the tensor evaluation quickly becomes complex and costly.

There is one special case, where the tensor reduces to a matrix - when f (or g by symmetry) can be evaluated at projection time (that means it is fixed). Then, we can define:

$$(M)_{i,j} = \sum_k (\int y_i(\omega) y_j(\omega) y_k(\omega) \mathrm{d}\omega) f_k \tag{3.9}$$

and

$$(fg)_i = \sum_j (M)_{i,j} g_j. \tag{3.10}$$

3.3 Properties of spherical harmonics

Spherical harmonics have certain interesting properties that make some algorithms more practical or that can speed up complex calculations. This lends to many applications in computer graphics, where functions defined over the sphere or hemisphere are very common.

There are mathematical operations, which may be performed more efficiently with functions given in terms of basis coefficients [Slo08]. For example, a convolution of a function f with circularly symmetric kernel k can be performed directly by multiplication of corresponding coefficients of the SH basis (up to a normalization constant):

$$(f * k)_l^m = K_l f_l^m g_l^0, \tag{3.11}$$

where

$$K_l = \sqrt{\frac{4\pi}{2l+1}}. \tag{3.12}$$

This property is analogous to the convolution theorem from the theory of Fourier analysis of ordinary 2D signals.

A direct consequence of spherical harmonics being an orthonormal basis is that we can very efficiently integrate a product of two functions. In this case, the product reduces to a simple dot product of the coefficient vectors and this property is the heart of *precomputed radiance transfer* ([SKS02], [KSS02]).

This technique aims at bringing some of the global illumination effects to real-time rendering, and uses the mentioned property to efficiently, but crudely, approximate the rendering equation. In its simplest form it computes the lighting as the integral of a product of precomputed BRDF and visibility information with incident lighting obtained at run-time. Both the lighting map and the BRDF+visibility information are stored as SH coefficients. This approach works only with isotropic BRDFs and therefore is not very practical, but there are extensions to this method which support even glossy BRDFs, but these are more complicated and beyond the scope of this thesis (see for example [PH10], pp. 974-982).

Spherical harmonics have another interesting property - rotational invariance. Suppose that we have SH coefficients of some function and we want to rotate this function on the sphere. This is for example required when we want to change the reference frame, in which the function is given. Typically, we might want to do some computation with BRDF or visibility information given in local space and with global lighting given in world space.

A straightforward method would be to rotate the original function and re-project to spherical harmonics, but we can do better - taking advantage of the rotational invariance, we can perform the rotation directly on the vector of coefficients. The invariance implies that we will get the same results.

Technically, the rotation is done by multiplying the coefficients vector with a special rotation matrix. A typical scenario is that we have a 3x3 matrix describing an ordinary spatial rotation and we want to convert it to a corresponding SH rotation matrix. This conversion can be derived analytically, but due to the complexity and computational inefficiency, most authors ([KSS02], [PH10] or [Slo08]) suggest decomposing the original 3x3 matrix into ZYZ Euler rotation matrices and doing three simpler SH rotations in succession. A mathematical derivation along with implementational issues is in [PH10], pp. 951-956. Even with this decomposition, the rotation procedure is too slow for many applications. For these purposes, an approximate SH rotation method was proposed in [KKB$^+$05].

3.4 Ringing

Spherical harmonics are not without limitations, though. The projection of a band-unlimited function to spherical harmonics will yield an infinite sequence of non-zero coefficients, which for practical purposes needs to be truncated. This step inevitably introduces errors to the reconstructed function.

Mathematically, truncated spherical harmonics expansion can be shown to be the minimizer of the least squares error functional:

$$\int_{4\pi} (f(\Omega) - \hat{f}(\Omega))^2 \, \mathrm{d}\Omega, \tag{3.13}$$

where

$$\hat{f}(\Omega) = \sum_{l=0}^{N} \sum_{m=-l}^{l} c_l^m y_l^m(\Omega))^2 \tag{3.14}$$

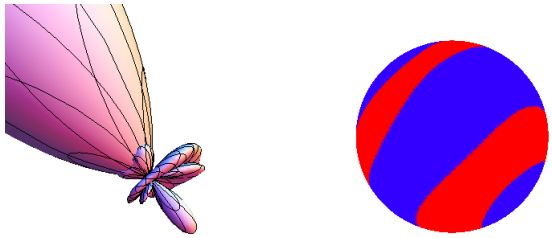


Figure 3.2: Left: An example of ringing artifacts in a reconstructed function. Before projection, the function was nonnegative and consisted only of the large lobe. Right: This visualization of the same reconstructed function shows that is has parts with negative values (shown in red).

Minimizing the square of the error allows the result to oscillate about the original function which gives rise to the so called ringing artifacts (figure 3.2 shows one example). Specially, for a strictly positive function f, its reconstruction $\hat{f}$ can have parts with negative values.

There are generally three techniques how to reduce this effect. The simplest method is to offset the function, so that negative values are avoided. This does not remove the ringing artifacts, though, it only removes one particular subproblem. Moreover, it is not clear how to find a suitable offset value. A general solution can be obtained either by *windowing* ([Slo08]) or by modifying the error minimizing functional 3.13 ([MHL09] or [Boy01], pp. 420-421).

Windowing is a very common technique in signal processing and intuitively means that we blur the function in the spatial domain to remove high frequencies, which can't be represented by the band limited SH representation. Blurring in the spatial domain can be described as a convolution with some filter (for example box filter). In the frequency domain, this amounts to multiplication of the SH coefficients with a kernel function that attenuates higher order coefficients. Sloan ([Slo08]) suggests using either the Lanczos function:

$$\frac{\sin \frac{\pi x}{N}}{\frac{\pi x}{N}} \tag{3.15}$$

or the Hanning function:

$$\frac{1 + \cos \frac{\pi x}{N}}{2}, \tag{3.16}$$

where x is the band we wish to attenuate and N is the order of SH representation we use.

The second approach is to use variational calculus and derive the conditions on coefficients that minimize some other functional than 3.13. A common technique is to add a regularization term that imposes some smoothness requirements on the minimizer. Boyd [Boy01] uses a penalizer in the general form:

$$\int_{4\pi} (\nabla^{2k} \hat{f}(\Omega))^2 \, \mathrm{d}\Omega, \tag{3.17}$$

while Sloan [Slo08] uses this term with $k = 1$, which then means we are minimizing the squared Laplacian of the reconstructed function. An analogy to physics can be drawn here - the Laplacian acts as a measure of energy and by minimizing it, we push the result to the lowest energy state possible, a constant function. The final functional is:

$$\int_{4\pi} (f(\Omega) - \hat{f}(\Omega))^2 \, \mathrm{d}\Omega + \lambda \int_{4\pi} (\Delta \hat{f}(\Omega))^2 \, \mathrm{d}\Omega \tag{3.18}$$

and its solution are coefficients in the form:

$$c_l^m = \frac{f_l^m}{1 + \lambda l^2 (l+1)^2}, \tag{3.19}$$

where f_l^m are coefficients of the original projection using 3.13 and λ is the regularization parameter. In [PH10], pp.950-951, the authors recommend setting $\lambda = 0.005$.

However, all these techniques change the shape of the reconstructed function in the sense that it becomes more uniform and smooth. This is an unwanted effect for our purposes, since we need to store our radiance estimates as accurately as possible. Moreover, none of the three presented approaches can guarantee a non-negative reconstruction $\hat{f}$ for an arbitrary non-negative function f and arbitrary maximum band N in general.

3.5 Sampling functions reconstructed from spherical harmonics

The simplest and most straightforward means of sampling is rejection sampling. Although definitely a valid method, it tends to be inefficient because it wastes many samples. Moreover, its performance is unpredictable since it depends on the particular function being sampled. However, the ability to generate the samples fast is crucial in the importance sampling process in the presented rendering algorithm.

Recently, a much more efficient strategy for importance sampling of functions given as spherical harmonics coefficients has been introduced in [JCJ09]. This has broadened the scope of applications of spherical harmonics to other fields of computer graphics. In our work, we utilize this sampling strategy in an unbiased Monte Carlo renderer.

The sampling scheme proposed by Jarosz et al. starts with a uniform sample distribution over the whole surface of the sphere and hierarchically warps it to match a prescribed function. During the warping step, the considered domain is split into four quadrants and integrals of the function over these sub-domains are computed. These four computed values serve as an importance function, which is used to warp the sample set. This step is then recursively repeated on the four quadrants.

Technically, the warping step is accomplished by doing a warp along the vertical axis first and then along the horizontal axis. For a domain T and its quadrants A, B, C, D (see Figure 3.3), this means we compute the integrals $I_1 = I_A + I_B$ and $I_2 = I_C + I_D$ of the reconstructed function and warp the set of the samples according to probabilities $p_{AB} = \frac{I_1}{I_T}$ and $1 - p_{AB}$. Warping along the horizontal axis is analogous. The effect of the warping step is that more samples are placed in areas with large values of $\hat{f}$. The whole process is visualized in figure 3.4.

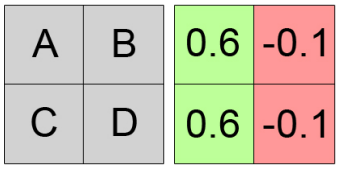


Figure 3.3: Left: definition of quadrants and integrals of the corresponding domains used in the text. For visualization purposes, we have mapped the spherical surface domain to a square. T denotes the union of all A, B, C, D. Right: one of the possible scenarios where some of the integrals are negative.

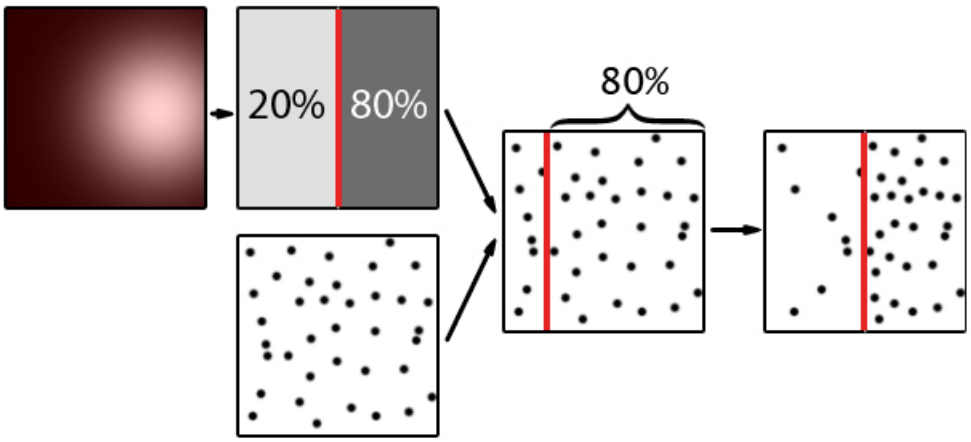


Figure 3.4: Illustration of one warping step. The leftmost square shows our function we wish to importance sample. First, the integrals of the function over the two subdomains are computed. In our case here, we find out that 80% of the function's mass lies in the right subdomain. These values are subsequently used to rescale the sample set to match the sampled function.

Warping continues in this fashion recursively up to a predefined maximum warping depth. The PDF of each sample is then computed from the ratio of the integral over the node containing the sample and the integral over the whole sphere.

This method generates samples that are distributed exactly proportionally to values of the reconstructed function $\hat{f}$ as long as the reconstruction is positive.

However, the importance sampling scheme of Jarosz et al. is particularly sensitive to the reconstruction artifacts described in the previous section, especially to the fact that the reconstructed function can have negative values. The hierarchical warping process used to generate the samples is undefined for negative

values (negative values can't be used to construct a valid PDF). Simple clamping of the negative values to zero will lead to bias as there will be parts of the function's domain which won't receive any samples.

The authors propose adding a positive offset to the function before projection, but finding a suitable value for this parameter automatically is an open problem. If the offset is set too high, it will prevent negative reconstruction issues but at the same time it will degrade the quality of the resulting distribution (it will tend towards globally uniform distribution).

On the other hand, some applications might not need a sample distribution that exactly matches the reconstructed function. An example of this is the rendering algorithm presented in the thesis, where we face the problem of importance sampling a local radiance estimate stored as a set of spherical harmonic coefficients. Here, the sampled function is inaccurate anyway, so an approximate sampling strategy is sufficient.

3.6 Our approach

Instead of trying to avoid negative reconstructed values completely, we use different rules during the warping process so that it can handle them in an unbiased way [Ber11].

The basic warping step is similar to [JCJ09]. First, the samples are warped along the vertical axis and then along the horizontal axis. As opposed to the original approach, we don't use the values of the integrals I_1, I_2, and corresponding probabilities $p_{AB} = \frac{I_1}{I_T}$, $p_{CD} = 1 - p_{AB}$ directly, but rather we use the values

$$\hat{p}_{AB}, \hat{p}_{CD} = 1 - \hat{p}_{AB} \tag{3.20}$$

, where

$$\hat{p}_{AB} \text{ is } p_{AB} \text{ clamped to the } [\epsilon, 1 - \epsilon] \text{ range} \tag{3.21}$$

for $0 < \epsilon \leq \frac{1}{2}$ (see figure 3.5). Warping along the second axis is analogous.

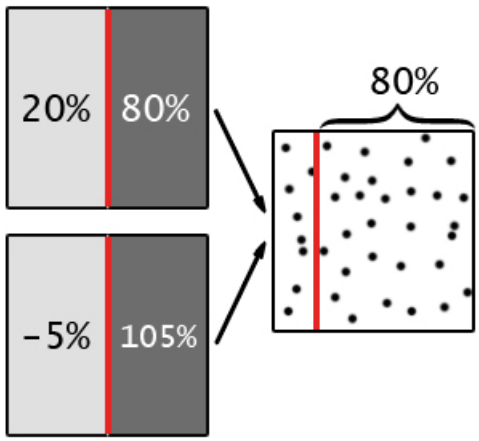


Figure 3.5: Our modified sampling strategy causes a whole range of integral values to be warped identically. In this case, $\epsilon = 0.2$.

Our observation is that this enables us to continue warping even if some of the integrals I_A, I_B, I_C, I_D are negative, but only as long as the total integral

I_T is positive. In effect, we modify the function we are trying to sample so that it has positive values of the respective integrals. If the total integral I_T is negative, we terminate the recursion immediately, which leaves the sample uniformly distributed in the domain of T as we have no suitable definition of corresponding sample distribution in this case.

Our scheme guarantees that we always get valid sample distributions and that there are no areas completely without any samples. This follows from the fact that at each warping level, the probability of each quadrant is at least ϵ^2, so for K levels of recursion, we have $p_X \geq \epsilon^{2K} > 0$ for all respective sub-regions X of $\hat{f}$. This along with the fact that we can compute the PDF of each sample exactly means that the importance function is nonzero over the whole domain and the Monte Carlo estimator remains unbiased for any $\epsilon \in (0, \frac{1}{2}]$.

3.7 Sample PDF

The PDF of each sample after the warping step can no longer be computed simply as the integral of the containing node divided by the total integral. This is because our clamping rule diverts the PDF of generated samples from the original function. Instead of the original calculation, we compute the final PDF incrementally during the recursion. Each warping step modifies the probability of a given quadrant from the original $\frac{1}{4}$ to $\hat{p}_h\hat{p}_v$ for the respective horizontal and vertical probabilities computed from $\hat{f}$. Therefore, we need to scale the sample PDF by the factor $\frac{\hat{p}_h\hat{p}_v}{\frac{1}{4}}$ for each warping level.

If we start with a PDF of a uniform distribution over the whole spherical domain, the final PDF of the sample (after k levels of warping) will be:

$$\frac{1}{4\pi}\prod_{l=1}^{k}\frac{\hat{p_h}\hat{p_v}}{\frac{1}{4}} = \frac{4^k}{4\pi}\prod_{l=1}^{k}\hat{p_h}\hat{p_v} \tag{3.22}$$

3.8 Error bounds

Let us consider the modified warping process. At each horizontal or vertical warping step, the probability clamping may cause the sample to end up in the other subdomain than in the original unmodified scheme with a probability bounded by ϵ. In other words, at most ϵ samples will be warped differently.

Then, because of the independency of the vertical and horizontal warps, the probability that a sample S at a particular recursion level will be warped differently than in the original scheme of Jarosz et al. is:

$$\begin{aligned} P(\text{S differs}) &\leq 1 - \bar{P}(\text{Vertical warp differs})\bar{P}(\text{Horizontal warp differs}) \\ &= 1 - (1-\epsilon)(1-\epsilon) \\ &= 2\epsilon - \epsilon^2. \end{aligned}$$

If we use k recursion levels, the warpings done at each of them are again independent and we get:

$$\begin{aligned} P(\text{S differs in } k \text{ levels}) &\leq 1 - \bar{P}(\text{S differs})^k \\ &= 1 - (1 - 2\epsilon - \epsilon^2)^k. \end{aligned}$$

For example, our used values of $k = 5$ and $\epsilon = 0.01$ yield at most 9.7% differently warped samples.

3.9 The role of ϵ

The value of ϵ generally affects the uniformity of the resulting distribution. Setting ϵ near zero will yield a distribution, whose PDF matches the original function very closely, but very few samples will be in the regions of negative reconstruction. In the limit case of $\epsilon = 0$, our method will return the same sample distribution as the original method of Jarosz et al. for functions which do not exhibit negative reconstruction issues.

On the other hand, setting $\epsilon = \frac{1}{2}$ will yield globally uniform distribution, as the probabilities will be equal in each warping step.

In our rendering system, where we sample functions that approximate local radiance estimates, we use a value of $\epsilon = 0.01$ so that the sample distributions match the functions closely.

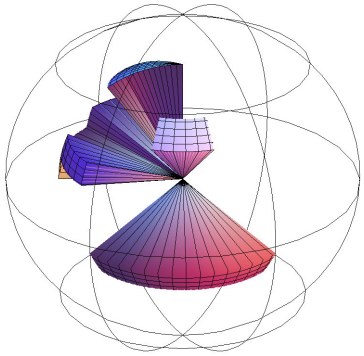


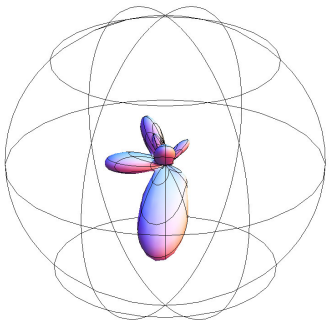
Figure 3.6: The original non-negative function (before projection) used for evaluation of our method. The blocky behavior and discontinuities are particularly difficult for spherical harmonics and severe ringing artifacts can be expected upon projection and reconstruction of this function.

3.10 Results

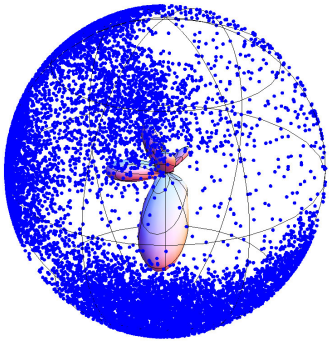
Figure 3.7 shows distributions obtained with our method and with the original method from [JCJ09] with offsetting. The same number of generated samples is shown for both methods. After reconstruction, our function from Figure 3.6 exhibits ringing artifacts and has parts with negative values. Note that function

offsetting causes the distribution to be much more uniform than the distribution from our method.

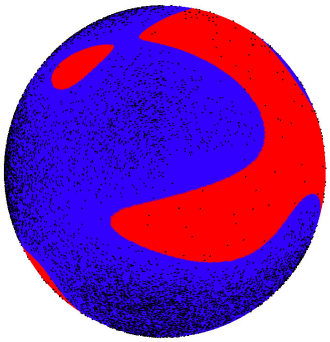
In our case, where we used the proposed method for importance sampling of local radiance estimates, the distribution generated with our method resulted in faster convergence, because fewer samples were sent to insignificant directions.



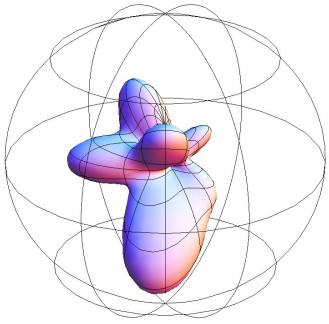
(a) The function reconstructed from projection to spherical harmonics using six bands.



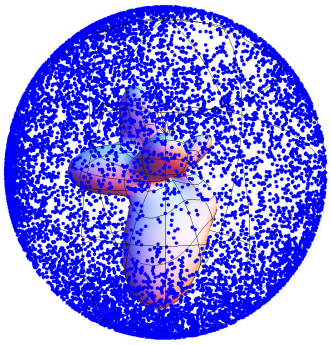
(b) Reconstructed function along with samples generated by our scheme with $\epsilon = 0.1$.



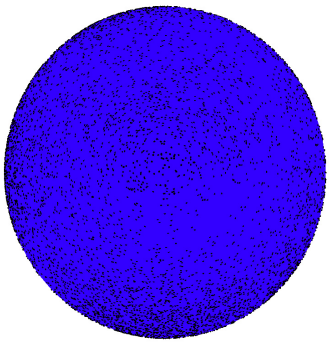
(c) Negative (red) and positive (blue) parts of the reconstructed function.



(d) Reconstructed function with offsetting. The minimum offset required to make the reconstruction positive across the whole spherical domain was determined by trial and error.



(e) Reconstructed function with offsetting along with samples generated by the original method of Jarosz et al.



(f) Negative (red) and positive (blue) parts of the reconstructed function.

Figure 3.7: A comparison of our method and the original method of Jarosz et al. The first row shows results obtained with our method. Note that the reconstructed function has large parts with negative values and that these regions do receive a fraction of the samples. On the contrary, to achieve non-negativity of the reconstructed function with the original method (the second row), a comparatively large offset value was needed, and the resulting distribution is much more uniform as a result.

Chapter 4

Guided path tracing

In this chapter, we propose an unbiased Monte Carlo rendering method based on path tracing. It combines previously published approaches in a way that offers significant performance improvements for difficult lighting scenarios such as caustics that are due to indirect lighting. However, our method is far more expensive in terms of computational requirements. To make the algorithm more practical, we propose a number of optimizations in chapter 5.

4.1 Overview

In the original path tracing algorithm [Kaj86] formulation, rays are traced throughout the scene by randomly selecting an outgoing direction at each bounce. This is, while mathematically correct, a very inefficient way of doing the traversal. According to Monte Carlo theory, an ideal solution would be to sample directions from a distribution that exactly matches the integrated function, which in this case amounts to the product of BRDF and incoming radiance.

The most common technique is BRDF importance sampling [LRR04], but it is inefficient if the BRDF is more or less diffuse, and the incoming radiance is highly non-uniform. Combined BRDF and incoming radiance sampling is a much less discussed option, but there are approaches in this direction, notably by Jensen in [Jen95]. In their approach, an estimate of incoming radiance is built at each bounce from a photon map created in a pre-process step. The photons are weighted by the respective BRDF values that depend on the ray outgoing direction, so the method indeed samples the product of BRDF and incoming radiance. However, these estimates can never be reused and need to be recreated at every bounce in order to support arbitrary (potentially anisotropic) BRDFs. This is unfortunately a very costly operation that makes the whole idea an impractical proposition.

Our proposed algorithm is a bit similar to that of Jensen [Jen95] in that it uses a photon map to obtain approximate incoming radiance estimates, but we use this information alone to do the importance sampling. Losing the dependency on the BRDF enables us to reuse the already computed estimates in hope of significantly increased performance. The main idea of our guiding method is shown in figure 4.1.

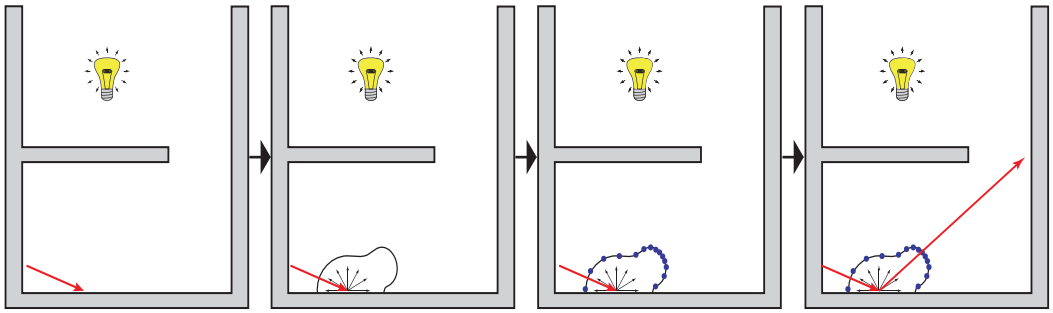


Figure 4.1: A visualization of the guiding process. From left to right: a ray hits a surface and the path tracer needs to determine the next outgoing direction. It queries the photon map for nearby photons and creates an approximation of the incoming radiance. In this case the incoming radiance function shows that there is a strong (indirect) illumination from the top right (due to light reflecting off the right side). A sample distribution is created based on this estimate so that more important areas receive more samples. Finally, a sample is drawn from this distribution and the recursion goes on.

4.2 Rendering algorithm

Like many improved versions of path tracing, our proposed method is a two-pass algorithm. In the first pass, we create a global photon map by tracing photons from the light sources and storing information for every bounce on non-specular materials. This step is similar to the shooting pass in Photon Mapping [Jen96]. The photons are stored in a balanced kd-tree, as suggested in the original paper or in [Jen04].

The second, gathering pass of our renderer is a hybrid path tracer that uses the information gathered during the photon tracing pass. As we shall see in chapter 5, the photon map is first used to determine the most suitable importance sampling strategy at each recursion level of a path, but here we will focus on how the photon map is used to obtain the rough radiance estimates and augment the importance sampling process.

It is important to note here, that our presented technique is aimed at improving the selection of the indirect lighting ray that continues the recursion, not on the rays that are used to gather direct lighting. For these, we use the same method as the classical path tracing algorithm - we use multiple importance sampling of BRDF and light sources. This technique performs well in a variety of lighting scenarios (see [VG95]).

To obtain the rough radiance estimates, we query the kd-tree and get a set of N closest photons to the given bounce point. These photons represent samples of the incoming radiance function we want to importance sample. For performance reasons, we do not directly query the photon map at each ray bounce, but instead we use an intermediate sparse data structure for pre-computation and caching of the incoming radiance samples. The samples are stored and created in a lazy fashion - whenever a sufficiently accurate sample can be obtained by interpolation from neighboring samples, the algorithm uses it. Only if the there are no suitable samples close to a given location, a new sample is built from the photon map and

subsequently stored in the cache.

A high level view of the operations done at each bounce is as follows:

1. Query the cache of samples for radiance estimate at the current position p.
2. If there are samples sufficiently close and the result can be computed by interpolation, use this estimate to sample a new direction.
3. Otherwise:
   (a) Query the photon map for radiance estimate at p.
   (b) Process the photons and create an incoming radiance function approximation.
   (c) Store the value to the cache and use it to sample a new direction.

4.3 Obtaining the irradiance estimates

Care must be taken so that the samples are represented in a way that allows fast interpolation and at the same time in a way that can be efficiently sampled. Our choice is to project the incoming radiance function to spherical harmonics basis and store a fixed number of the resulting coefficients. Interpolation of the projected functions then amounts to a simple interpolation of spherical harmonic coefficients. Also, spherical harmonics can be efficiently and robustly sampled, as shown in chapter 3. The quality of the irradiance function representation mainly depends on the number of SH bands used (see figure 4.2). In our tests, we generally use 8 bands.

Spherical harmonics are not without limitations, though, as already described in chapter 3. They are prone to ringing artifacts and they are not particularly good at representing highly discontinuous functions, especially when a low number of bands is used. There are two other bases we have considered as well: Slepian functions [SDW06] and $\mathcal{H}$-basis [HW10]. Both provide a theoretically better set of basis functions (related to spherical harmonics) for representing functions on a spherical subdomain. However, there is no published efficient sampling method for them, so we would have to use rejection sampling, which would make the whole rendering scheme impractically slow.

When a new sample is to be computed, the incoming directions of photons obtained from the photon map are projected to the spherical harmonics basis. This is accomplished by summing up the contributions of all photons according to equation 3.5. For each photon, all basis functions need to be evaluated at the photon's incoming direction and multiplied by the radiance carried by the photon.

In some cases, especially if there are too few photons shot in the pre-processing phase, this irradiance estimate can be inaccurate. The sampling scheme guarantees that the algorithm remains unbiased even in these cases (see chapter 3 or [Ber11]), but the convergence speed can be decreased as a result. We have found that in these cases, it might be beneficial to use a higher value of ϵ (see 3.9) to make the sampling distributions more uniform.

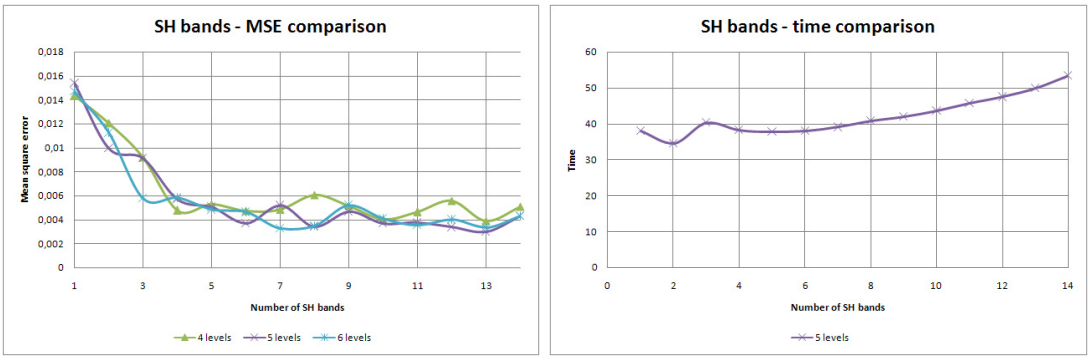


Figure 4.2: A plot of rendering results for different numbers of spherical harmonics bands. The left plot shows how, for a given scene and number of samples per pixel, the MSE varies with respect to the number of SH bands. The number of levels for the three different lines refers to the number of levels in the hierarchical warping during spherical harmonics sampling process (see [JCJ09] for details). Similarly, the right plot shows the dependence of running time on the number of SH bands used. These two plots suggest that the optimal choice for the number of SH bands is around 8 for 5 warping levels.

4.4 Storing the incoming radiance estimates

For scenes resembling real world settings, the incoming radiance function exhibits significant spatial coherence because many objects are at least partly formed by locally flat surfaces, where the incident light varies slowly. Drawing from this observation, it seems that it could be beneficial to cache the already computed estimates and create new ones by interpolation. Interpolation of SH coefficients is a much less expensive operation than photon map lookup and SH projection, so a significant speed-up can be expected.

This idea is not new - it is the central concept of irradiance caching, a rendering algorithm suited mainly to diffuse-like surfaces whose roots date back to 1988 [WRC88], and which has been since extensively studied and improved [KGW+08]. The algorithm computes irradiance values by stochastic integration of the incoming radiance over the full hemisphere. These values are in turn stored in a spatial data structure (the authors suggest a variation of an octree) along with a validity radius computed as a clamped distance to nearest surface. Whenever a new irradiance value is required, the algorithm first looks into the cache and tries to interpolate neighboring samples. A combination of heuristics is used to weigh the samples - the two most important are distance and surface normal difference, which cause samples farther away or samples on differently oriented surfaces to have less influence on the result. If no suitable samples are found, a new irradiance value is computed by integration and stored in the cache.

Most of the features and implementation details can be immediately used for our purposes, but there are certain aspects, which are different in our case. The most notable one is that we cannot completely rely on the spatial coherence assumption because we need to capture subtle lighting details as well. Caustics, shadows and similar lighting conditions result in abrupt changes in the incoming radiance function, so in addition to nearest surface distance we introduce another heuristic that influences the validity region of each cache entry. It is based on

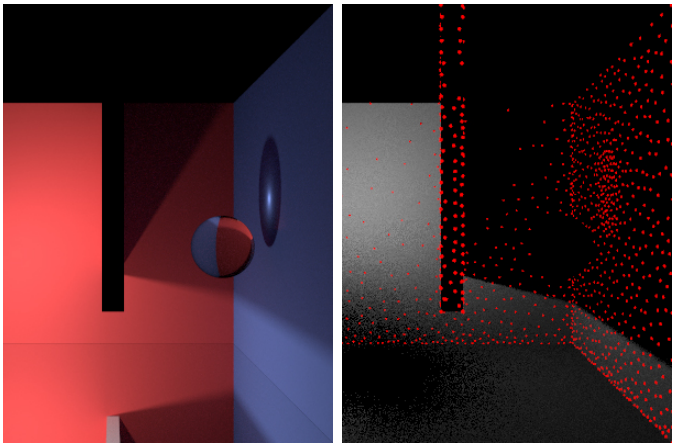


Figure 4.3: A visualization of the incoming radiance function cache. The left image shows the scene rendered with complete global illumination. A small spherical light is located to the left (outside the images). The right image shows the samples that were created by our algorithm as red dots. Note how the heuristics cause the samples to vary in their density, especially near the caustic and along corners.

the *causticness* value, which is defined and described in chapter 5. This heuristic causes the cache to create the samples more densely in regions with caustics. Figure 4.3 shows the distribution of the cache entries in a test scene after the image was rendered and the cache was filled.

Chapter 5

Optimizations

The main contribution of this chapter is a modification of our rendering algorithm that is capable of efficiently switching its sampling strategy. The two strategies it can choose from are regular path tracing (RPT), which samples the BRDF to determine next recursion ray, and incident radiance importance sampling (IRIS) presented in chapter 4. IRIS is clearly, on a per-path basis, much slower than RPT for most types of path because of the overhead associated with spherical harmonics projection, sampling and caching. But there are certain illumination conditions when IRIS is still a significantly superior sampling strategy, so it does make sense to use it in some circumstances. This suggests that a renderer which is capable of switching between them might be worth investigating.

The idea is to adaptively switch between the strategies in a way that the overhead caused by the switching process does not cancel out the gains obtained by using the "right" technique at each recursion level of a given path. To this end, we introduce a framework for quickly identifying those parts of a rendered scene where "difficult" lighting conditions prevail, and where the more computationally costly (but in those cases also more efficient) IRIS ought to be used, instead of BRDF sampling. Scenarios where IRIS should be invoked are generally areas with a strong indirect illumination via specular surfaces, and in particular caustics.

5.1 Causticness

We need to be able to decide the sampling strategy prior to actually using it - we cannot leave IRIS switched on "just in case", to be e.g. weighed against the results of RPT later. Therefore, we developed a method which is capable of rapidly estimating how hard the rendering would be for regular path tracing using RPT at any given surface point in the scene. We call this measure *causticness*, and we compute it from additional numeric value that is stored at each bounce in the photon map during the photon shooting step. This value for each photon is initialized to 1 and at each bounce, it gets multiplied by a constant c, which is different for every BSDF type. These constants are set so that specular reflections and refractions increase the causticness value significantly ($c = 1.4$ for reflections and $c = 2$ for refractions), while diffuse bounces reduce it ($c = 0.1$). This ensures that photons which are likely to belong to a caustic will have a large causticness. For visualization of causticness computed for different scenes, see figure 5.1.

During the path tracing step, this causticness value is examined at each bounce

to determine if IRIS should be used to sample the next direction. To obtain the causticness value for a given point, we simply average the causticness values of nearby photon hits. If the causticness is below some threshold, the path tracer falls back to plain BRDF sampling, which handles simple lighting situations well. To provide fast access to the causticness estimates, we bake the values to texture maps in a pre-process step just after building of the photon map and sample them at runtime.

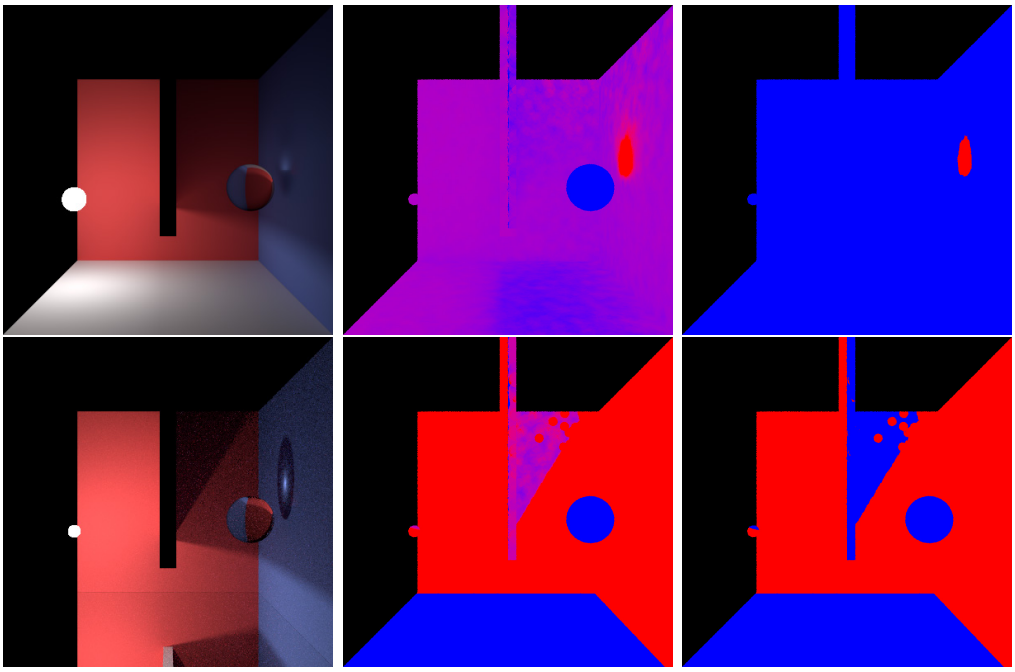


Figure 5.1: Visualization of the computed causticness values. The left column shows scenes for which the causticness was computed. The first row contains a scene with a diffuse floor, whereas floor on the images in the second row is a mirror.
The middle column shows a visualization of the computed causticness values. Red color shows parts with a high causticness value. The third column shows where our guided path tracer was used (red parts) and where a regular path tracer was used (blue parts).

Chapter 6

Implementation

6.1 Programming language and the environment

We have implemented the described algorithm along with the optimizations in the Physically based rendering toolkit (PBRT), version 2 [PH10]. This decision allowed us to focus on our algorithm itself and not on features common to every renderer such as I/O, scene representation, ray-geometry intersection and so on. Furthermore, PBRT contains implementations of other rendering algorithms, which can be used either directly as parts of our algorithm (for example photon mapping or irradiance caching) or for performance comparison (MIS path tracing and Metropolis light transport).

PBRT is written in C++ and is designed to be cross-platform and multi-threading enabled. During our development, we have successfully compiled and run it on Microsoft Windows as well as Linux machines with up to 12 processor units. We have found that compiling PBRT as a 64-bit application results in significantly increased performance and all our tests were run with a 64-bit executable. Our tests showed that on average, the 64-bit version performs 20-30% better than the 32-bit version. An important aspect of the library is that it is primarily intended as a research and teaching tool, not as a production renderer. The program is well structured and thoroughly described in the accompanying book.

The following sections describe how our algorithm is implemented in PBRT and some other important implementation issues.

6.2 Implementation overview

Most of the functionality of our algorithm is in the `PhotonPathIntegrator` class. In the PBRT rendering framework, it is technically a *surface integrator* module (see [PH10], pp. 739-741), which is responsible for computing radiance along a given ray. The primary rays are spawned in other modules - the surface integrator only takes them and computes both direct and indirect illumination. As a result, it outputs the computed radiance values. The most important methods are `Preprocess` and `Li`.

6.3 Pre-processing

In the `Preprocess` method, we first create the photon map. This step is similar to the original PBRT implementation, but there are a few differences. We do not separate direct, indirect and radiance photons, but we store only indirect photons in one global photon map instead. The reason for this is that we are interested only in indirect lighting for our radiance estimates. Direct lighting is in our case handled by light source sampling and thus should not be included in our estimates. A second major modification is that we compute and store the causticness values for each photon as it is traced throughout the scene. The relevant code is contained in the `PPPhotonShootingTask` class.

The next thing to do, once the photon map is created, is to bake the causticness values to texture maps. This step is an important optimization, since it removes the burden of expensive and frequent photon map queries at runtime and replaces it with fast 2D texture lookups. A disadvantage of this approach might be that the geometry must have valid texture coordinates. However, this is not a big limitation in practice because geometric models exported from typical 3D modeling software usually already have them. If they are not present, our algorithm skips the texture generation and falls back to photon map queries at runtime. During the baking step, we encountered an interesting problem of converting sampled UV coordinates to world space position of the geometry model. Our implementation (`TriangleMesh::GetWorldPosition`) is based on [Cha03] and calculations in [Hec 6].

The last task of the `Preprocess` method is to precompute integrals of spherical harmonics basis functions over various domains (see the optimization part of [JCJ09]).

6.4 Rendering

The core of our algorithm is implemented in the `Li` method. It receives a ray as its parameter and computes the radiance exiting the scene along it. In a loop, it traces the given ray through the scene in a way similar to the path tracing algorithm. At each bounce, the algorithm computes direct lighting contribution by BRDF and light source sampling combined according to the Multiple importance sampling method. Then, it examines the surface hit by the ray to determine the method of choosing the direction of the next indirect lighting ray. There are several cases that are handled differently:

- The surface is specular (or more precisely, non-diffuse). Since specular BRDFs are handled well by BRDF sampling, the algorithm uses it to sample the next direction.
- The surface is diffuse. The algorithm computes the causticness value and uses it to decide between BRDF sampling and incident radiance sampling.
- The BRDF of the surface has several components, some of which are diffuse and others not. In this case, the algorithm probabilistically chooses one of the components and continues as in the previous two cases.

If the algorithm decides to use the incident radiance sampling, it follows the steps described in 4.2.

6.5 SH sampling

Our implementation is directly based on the algorithm described in [JCJ09] and our modifications described in chapter 3. It is vital to implement all optimizations and precomputations suggested in the original paper to achieve the proclaimed performance. The implementation of most precomputations is in the `shsampler.cpp` file. The actual sampling is written in the `warpSampleNonRecursive` method of the `PhotonPathIntegrator` class.

6.6 Concurrency issues

The implementation of PBRT executes all performance critical computations with multiple threads in parallel. In our case, there are two major tasks that are parallelized: photon shooting and ray traversal. As our implementation of the photon shooting step is based on PBRT code base, it does not need any special adjustments in this regard.

However, this is not the case for the second task - ray traversal. The octree cache of SH estimates poses an additional structure that is shared among the threads and which needs to be managed to prevent data corruption.

Our first implemented mechanism was a mutex along with critical sections locking. Although this solution was correct in the sense that it prevented the threads from simultaneous read/write access to the cache, careful debugging and profiling on a multi-core machine revealed an important inefficiency. After the cache had been sufficiently filled (especially if the samples density was set to be relatively high), and accesses to the cache had become more time consuming, most of the threads were stalled waiting for one particular thread to finish writing to the cache. This caused the performance to severely degrade over time.

To circumvent this effect, we replaced the mutex with a wait-free implementation described in [DDdSC11], which does not require locking mechanisms and relies on two special CPU instructions offered by modern architectures. These are compare and swap (CAS), and fetch and add (XADD). Our implementation is in the `octree.h` file.

Chapter 7

Performance analysis

We implemented the described technique in the PBRT rendering research framework. The results obtained with our algorithm are compared against PBRT's implementation of path tracing (with multiple importance sampling) and Metropolis light transport [VG97].

7.1 Tests and results

To evaluate our algorithm, we did test runs on four different scenes. Three of them are artificial and targeted at specific illumination conditions, while the fourth scene is a model of a villa from the PBRT scenes package.

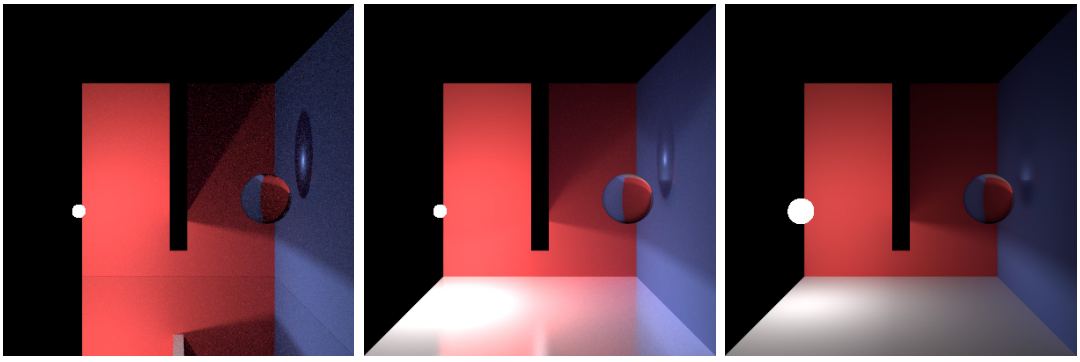


Figure 7.1: Scenes used to evaluate the performance of the proposed method. The scenes differ only by the floor material and light source size. Left: perfect mirror, Middle: glossy floor, Right: Lambertian floor.

The three artificial test scenes are shown in figure 7.1. They all contain similar geometric setting, where a glass sphere is lit only by indirect illumination. This is accomplished by putting a blocker object between the light source and the glass sphere. The glass sphere causes indirect light that reflects on the floor to refract and create a caustic pattern on the wall behind the sphere. The floor material determines the appearance of the caustic - it is more blurry for diffuse floor and sharp for a mirror.

The most difficult scenario is the first one (figure 7.1, Left). Only very few rays from the caustic eventually hit the light source if regular BRDF sampling is used. The reason for this is that the light source occupies only a very narrow solid

angle and BRDF sampling is likely to miss it because it samples the hemisphere of outgoing directions uniformly for diffuse BRDFs. Even the Metropolis renderer has difficulties to render the caustic. On the other hand, our proposed algorithm speeds up convergence considerably already for moderate numbers of samples per pixel, because the respective incoming radiance estimates show a peak in the direction of the light source and the sample distribution is proportional to the estimates.

The right image in figure 7.1, on the other hand, contains a scenario which is handled well by the regular BRDF and light source sampling. Here, much larger areas and solid angles contribute to the blurry caustic, so the benefits of using our sampling scheme are outweighed by the increased computing cost here. But thanks to our decision strategy based on causticness estimates, our renderer reverts to regular BRDF sampling and the results are nearly identical to the path tracer. The small difference is due to the increased startup cost (creating the photon map and causticness maps).

The real world scene shows the interior of a villa model lit by indirect lighting from the outside (see figure 7.2). It consists of all kinds of objects, but there are no particularly difficult lighting conditions like caustics. Although better than regular path tracing, our algorithm performs worse when compared to Metropolis light transport. This is partly because the photon shooting step is rather costly here and partly because there are no very difficult parts, where our algorithm would be beneficial.

Figure 7.2: The villa scene used to evaluate the performance of the proposed method.

Plots in figures 7.3, 7.4, 7.5 and 7.6 show the numerical results of all three algorithms obtained on a compute server with 12 Opteron cores. Our algorithm was set to 8 SH bands and 5 levels of warping during SH sampling. We used 500000 photons for the three artificial scenes and 5000000 photons for the villa scene. Also note that all scales are logarithmic.

For the specific lighting conditions created in the first three scenes, we see that our algorithm performs similarly or better than path tracing and Metropolis light transport. However, for the villa scene, we clearly see the efficiency of

the Metropolis renderer, which handles the scene much better than both our algorithm and standard path tracing.

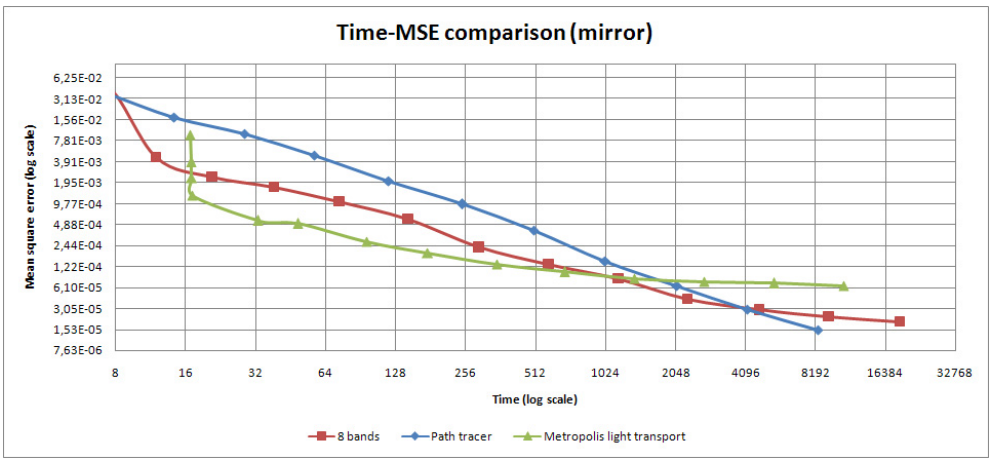


Figure 7.3: A plot of time—MSE dependency for the artificial light setting with a mirror floor.

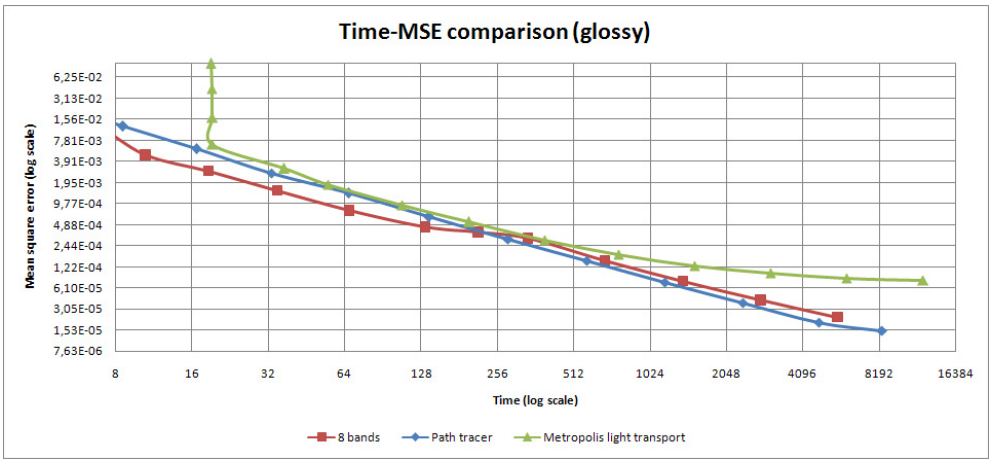


Figure 7.4: A plot of time—MSE dependency for the artificial light setting with a glossy floor.

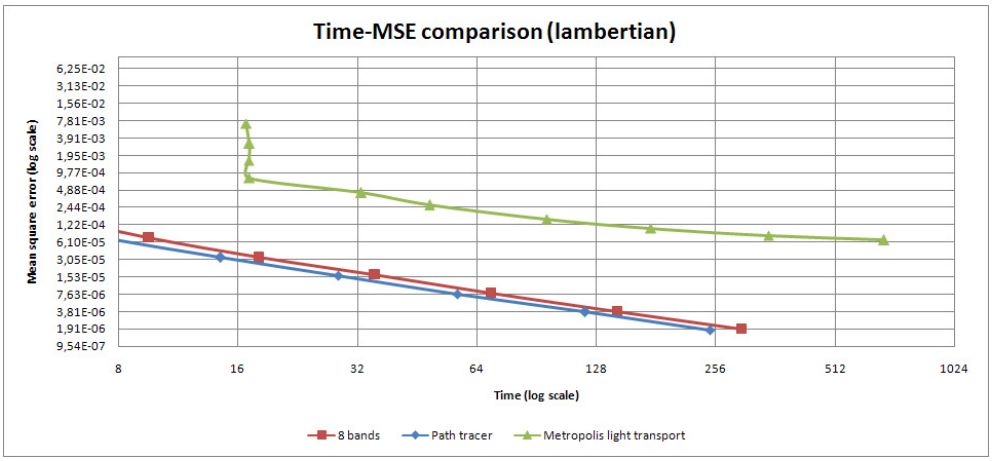


Figure 7.5: A plot of time—MSE dependency for the artificial light setting with a diffuse floor.

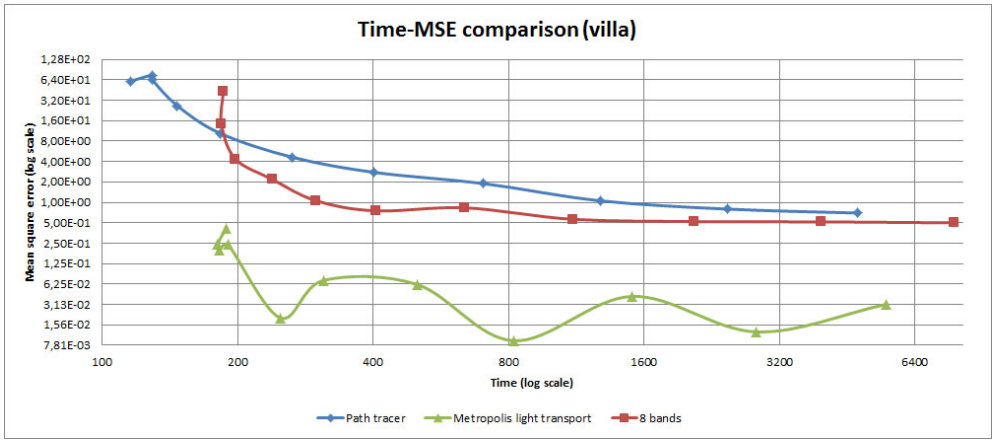


Figure 7.6: A plot of time—MSE dependency for the villa scene.

Chapter 8

Conclusion

8.1 Summary

In the thesis, we have introduced the main concepts and challenges of unbiased rendering. After a brief introductory part, we have described the rendering equation, which is a central concept of unbiased rendering and provides an elegant framework for theoretical studies of rendering algorithms. Following a detailed description of the path tracing algorithm, the first algorithm specifically designed for solving the rendering equation, we have summarized other currently available algorithms along with their advantages and limitations.

Then, we have proposed a novel two-pass rendering algorithm based on path tracing. In the first pass, we gather information about the light distribution in the scene by shooting and tracing photons from the light sources and storing information along their paths. Then, in the second pass, we use the data gathered during the first pass to aid the path tracer during importance sampling. This is accomplished by reconstructing approximate local irradiance estimates and using them to build the sample distributions. The irradiance estimates are computed on demand from the photon map and are stored as an array of spherical harmonics coefficients in a sparse data structure that can be used to obtain the estimates quickly by interpolation. The proposed sampling approach offers interesting convergence improvements for difficult lighting scenarios, where other methods are inefficient.

Moreover, to amortize the cost of using a more complicated sampling approach, we propose a measure of lighting condition "difficulty" called *causticness*. The cost of computing it in a pre-processing step is negligible and at run-time, the renderer uses it to dynamically switch between our proposed sampling strategy and regular path tracing.

We also provided some mathematical background of spherical harmonics, which are one of the key internal features of our approach. We described a previously published technique for importance sampling spherical harmonics along with an analysis of the problems which arise during SH reconstruction.

Another original contribution of the thesis is a modified approach of sampling functions given in terms of spherical harmonic coefficients, which is robust in the presence of reconstruction errors. The distribution generated with our method is exactly proportional to the sampled function in the regions without reconstruction issues and is uniform in the problematic regions. Also, there is virtually no

memory requirements or performance penalty associated with our modifications.

We have evaluated our method in a variety of lighting conditions and we have presented a comparison against other rendering methods.

8.2 Fulfillment of the goals

We have developed a working rendering algorithm that uses the idea of improved path tracing. We have presented an analysis of the algorithm and we have shown a performance comparison with regular path tracing and the Metropolis light transport algorithm. Our results demonstrate the viability of the approach even though we are not able to deliver results which would beat other algorithms consistently. We have attached a sample implementation in the PBRT library and used exactly this implementation to generate our results.

The algorithm remains unbiased, which is a consequence of it being based on path tracing, which is an unbiased algorithm, along with our observations and improvements of the spherical harmonics sampling method.

8.3 Future directions and discussion

Our algorithm and its analysis show that the idea of augmented path tracing might result in significant improvements. Given the same number of samples per pixel, our algorithm is, depending on the scene, capable of rendering images with an order of magnitude better MSE. However, this comes at the cost of substantially slower path throughput. The figure a typical user is interested in is the Time/MSE ratio. In humble words, the user has a fixed amount of rendering time and needs the best image possible in terms of MSE.

Our proposed algorithm is advantageous for specific difficult light transport scenarios, and in those cases it is even capable of beating the Metropolis light transport algorithm. However, for a real world setting which consists of many mostly diffuse and glossy objects, the increased time price for the superior importance sampling strategy overweighs the gains in MSE. Our causticness optimization does improve the situation, but there is certain (in practice not negligible) overhead associated with its computation. Additional overhead comes from the photon shooting pass.

An interesting possibility for future work is to use our sampling scheme in the first stage of Metropolis light transport, where the algorithm searches for important light transport channels by bi-directional path tracing. By utilizing our strategy, it could be able to find those channels more quickly and then explore them by the unmodified second stage.

To keep the speed of our sampling method reasonable, we completely disregard BRDF information and rely solely on the incoming radiance estimates. This is certainly not the optimal strategy and it can be another direction of future work to combine BRDF and incoming radiance sampling.

This would require storing the BRDF in a form similar to our radiance estimates, so spherical harmonics are obviously the first candidate. Another possibility would be to use for example Haar wavelets. The main difficulty is how to combine the two estimates and sample their product. As we have seen in chapter

3, evaluating a product of two SH functions is a non-trivial task involving difficulties like computationally expensive tensor multiplication. Furthermore, we keep our radiance estimates in world space to enable their interpolation, whereas the BRDF would need to be stored in local object space. This suggests that prior to the multiplication, one of the functions would need to be rotated, which is again a non-trivial task.

The published sampling approaches suggest that it might be possible to sample the product of the two functions without actually evaluating it ([JCJ09, CAM08]). We have not investigated this possibility further and it is also left for future work.

Appendix A

Contents of the accompanying DVD

The structure of the accompanying DVD is as follows:

- `/Binary` - contains pre-built Windows binaries of PBRT with our implementation. Both the 32-bit and 64-bit versions are included. User's guide can be found in appendix B of the thesis.
- `/Source` - contains the complete source code of the PBRT library with our implementation. Also contains the test scenes in the /Scenes subfolder.
- `/Thesis` - contains the *pdf* file and LaTeX source code of this thesis.

Appendix B

User's guide

Our algorithm can be switched on in the PBRT scene definition file by specifying the `photonpath` integrator and by properly configuring the following parameters. All float parameters can take on arbitrary positive values, if not specified otherwise.

Parameter	Description	Default
`integer nused`	Maximum number of photons that are used to build the incoming radiance estimates. The actual number of photons used may be smaller than the specified value depending on the photon density around the queried locations.	50
`float maxdist`	The radius of validity for each photon.	0.1
`float shoffset`	An offset value that is uniformly added to all SH estimates to prevent ringing and reconstruction errors.	0.0
`float minsampleextent`	Minimum extent of validity for SH estimates in the octree cache structure.	0.05
`float maxsampleextent`	Maximum extent of validity for SH estimates in the octree cache structure.	0.5
`float minweight`	Minimum total interpolation weight that the interpolated SH estimate must have in order to be accepted. If the computed value is less than this threshold, a new SH estimate is built from the photon map.	0.2
`float maxangledifference`	Maximum allowed difference of surface normals for SH estimated interpolation.	10.0

Parameter	Description	Default
`float probepsilon`	The value of ϵ used during SH sampling (see section 3.9). The range of allowed values is $(0, 0.5]$.	0.0001
`float causticnessthreshold`	The causticness threshold used to determine the sampling strategy at each bounce.	0.2
`float causticnessweight`	Determines the strength of causticness influence on the spacing of samples in the cache of SH samples. Lower values mean greater influence.	0.1
`bool causticnessvisualization`	Setting this value to true will output a red/blue binary image, which can be used to identify the sampling strategy at all parts of the scene. Red value signifies our proposed method, blue means conventional BRDF sampling.	false

A whole surface integrator block with all mentioned settings can look as follows:

```
SurfaceIntegrator "photonpath"
   "integer nused" [120]
   "float maxdist" [ .05]
   "integer photons" [500000]
   "float shoffset" [0.0]
   "float minsampleextent" [0.05]
   "float maxsampleextent" [0.5]
   "float minweight" [0.5]
   "float maxangledifference" [10.0]
   "float probepsilon" [0.0]
   "float causticnessthreshold" [0.2]
   "float causticnessweight" [0.1]
   "bool causticnessvisualization" ["false"]
```

To enable the causticness optimization, where the values are baked to texture maps (introduced in section 5.1), it is important that the geometric models have texture coordinates. If they are not present, the algorithm will still work, but will query the causticness value from the photon map, which is a slower operation than texture lookup.

Bibliography

[Ber11] Martin Berger. Approximate importance sampling of functions reconstructed from spherical harmonics. In Vaclav Skala Gladimir Baranoski, editor, *Communication Papers Proceedings of WSCG 2011*, page 181, February 2011.

[Boy01] J. P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications, New York, 2001.

[CAM08] Petrik Clarberg and Tomas Akenine-Möller. Practical Product Importance Sampling for Direct Illumination. *Computer Graphics Forum (Proceedings of Eurographics 2008)*, 27(2):681–690, 2008.

[CBNR11] Manmohan Chandraker, Jiamin Bai, Tian-Tsong Ng, and Ravi Ramamoorthi. On the duality of forward and inverse light transport. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[Cha03] Keshav Channa. Light mapping - theory and implementation, July 2003.

[CPC84] Robert L. Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '84, pages 137–145, New York, NY, USA, 1984. ACM.

[DBB06] Philip Dutré, Kavita Bala, and Philippe Bekaert. *Advanced global illumination, 2nd edition*. A K Peters (http://www.akpeters.com/), 2006.

[DDdSC11] Kurt Debattista, Piotr Dubla, Luis Paulo Peixoto dos Santos, and Alan Chalmers. Wait-free shared-memory irradiance caching. *IEEE Computer Graphics and Applications*, 31:66–78, 2011.

[Gre03] R. Green. Spherical harmonic lighting: The gritty details. *Archives of the Game Developers Conference*, March 2003.

[Hec 6] Chris Hecker. Perspective texture mapping. *Game Developer*, 1995-6.

[HJ09] Toshiya Hachisuka and Henrik Wann Jensen. Stochastic progressive photon mapping. *ACM Transactions on Graphics*, 28(5), 2009.

[HOJ08] Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. *ACM Transactions on Graphics*, 27(5), December 2008.

[HW10] Ralf Habel and Michael Wimmer. Efficient irradiance normal mapping. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, I3D '10, pages 189–195, New York, NY, USA, 2010. ACM.

[JCJ09] W. Jarosz, N. A. Carr, and H. W. Jensen. Importance Sampling Spherical Harmonics. *Computer Graphics Forum (Proc. Eurographics EG'09)*, 28(2):577–586, 4 2009.

[Jen95] Henrik Wann Jensen. Importance driven path tracing using the photon map. In *Eurographics Rendering Workshop 1995*. Eurographics, June 1995.

[Jen96] Henrik Wann Jensen. Global illumination using photon maps. In Xavier Pueyo and Peter Schröder, editors, *Eurographics Rendering Workshop 1996*, pages 21–30, New York City, NY, June 1996. Eurographics, Springer Wien.

[Jen04] Henrik W. Jensen. A practical guide to global illumination using ray tracing and photon mapping. In *SIGGRAPH '04: Proceedings of the conference on SIGGRAPH 2004 course notes*, New York, NY, USA, 2004. ACM Press.

[Kaj86] James T. Kajiya. The rendering equation. In *Proceedings of Siggraph '86*, pages 143–150, 1986.

[KGW+08] Jaroslav Křivánek, Pascal Gautron, Greg Ward, Henrik Wann Jensen, Per H. Christensen, and Eric Tabellion. Practical global illumination with irradiance caching. In *ACM SIGGRAPH 2008 classes*, SIGGRAPH '08, pages 60:1–60:20, New York, NY, USA, 2008. ACM.

[KKB+05] Jaroslav Křivánek, Jaakko Konttinen, Kadi Bouatouch, Sumanta Pattanaik, and Jiří Žára. Fast approximation to spherical harmonic rotation. In *SCCG '06: Proceedings of the 22nd spring conference on Computer graphics*, New York, NY, USA, 2005. ACM Press.

[KSS02] J. Kautz, P. P. Sloan, and J. Snyder. Fast, arbitrary brdf shading for low-frequency lighting using spherical harmonics. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 291–296, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[LRR04] Jason Lawrence, Szymon Rusinkiewicz, and Ravi Ramamoorthi. Efficient BRDF importance sampling using a factored representation. In *ACM SIGGRAPH 2004*, August 2004.

[LW93] Eric P. Lafortune and Yves D. Willems. Bi-directional Path Tracing. In H. P. Santo, editor, *Proceedings of Third International Conference on Computational Graphics and Visualization Techniques (Compugraphics '93)*, pages 145–153, Alvor, Portugal, December 1993.

[MHL09] R. G. McClarren, C. D. Hauck, and R. B. Lowrie. Filtered spherical harmonics methods for transport problems. In *Proceedings of the International Conference on Mathematics and Computational Methods and Reactor Physics, American Nuclear Society*, 2009.

[PH10] Matt Pharr and Greg Humphreys. *Physically Based Rendering, 2nd ed.* Morgan Kaufman, 2010.

[PM95] S. N. Pattanaik and S. P. Mudur. Adjoint equations and random walks for illumination computation. *ACM Trans. Graph.*, 14:77–102, January 1995.

[PTVF92] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical recipes in C (2nd ed.): the art of scientific computing.* Cambridge University Press, New York, NY, USA, 1992.

[SDW06] Frederik J. Simons, F. A. Dahlen, and Mark A. Wieczorek. Spatiospectral concentration on a sphere. *SIAM Rev.*, 48:504–536, March 2006.

[SKS02] P. P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, 21(3):527–536, 2002.

[Slo08] P. P. Sloan. Stupid spherical harmonics (sh) tricks. *Game Developers Conference*, February 2008.

[Vea98] Eric Veach. *Robust monte carlo methods for light transport simulation.* PhD thesis, Stanford, CA, USA, 1998. AAI9837162.

[VG95] Eric Veach and Leonidas J. Guibas. Optimally combining sampling techniques for monte carlo rendering. In *Computer Graphics Proceedings, Annual Conference Series, 1995 (SIGGRAPH 95)*, pages 419–428, August 1995.

[VG97] Eric Veach and Leonidas J. Guibas. Metropolis light transport. In *Computer Graphics (ACM SIGGRAPH '97 Proceedings)*, volume 31, pages 65–76, 1997.

[WRC88] Gregory J. Ward, Francis M. Rubinstein, and Robert D. Clear. A ray tracing solution for diffuse interreflection. *SIGGRAPH Comput. Graph.*, 22:85–92, June 1988.